EXHIBIT A

IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF WISCONSIN

MOTOROLA MOBILITY, INC., and GENERAL INSTRUMENT CORPORATION, Plaintiffs, v. MICROSOFT CORPORATION, Defendant.	Civil Action No. 3:10-cv-699 JURY TRIAL DEMANDED
MICROSOFT CORPORATION, Counterclaim Plaintiff, v. MOTOROLA MOBILITY, INC., and GENERAL INSTRUMENT CORPORATION, Counterclaim Defendants.	

DEFENDANT MICROSOFT CORPORATION'S ANSWER AND COUNTERCLAIMS TO PLAINTIFFS' FIRST AMENDED COMPLAINT FOR PATENT INFRINGEMENT

Microsoft Corporation ("Microsoft") answers Motorola Mobility, Inc.'s ("Motorola Mobility") and General Instrument Corporation's ("General Instrument") (collectively, "Motorola") First Amended Complaint for Patent Infringement ("Complaint"), filed on January 11, 2011, as follows (the numbered paragraphs below correspond to the like numbered paragraphs in the Complaint and any allegation of the Complaint not expressly admitted below is denied):

NATURE OF THE ACTION

1. Microsoft admits that Motorola has brought this action alleging infringement by Microsoft of United States Patent Nos. 7,310,374 ("the '374 Patent"); 7,310,375 ("the '375 Patent"); and 7,310,376 ("the '376 Patent") (collectively, "the Motorola Asserted Patents"). However, Microsoft denies committing any infringement or other tortious or unlawful act and denies that Motorola is entitled to any remedy for Microsoft's actions.

THE PARTIES

2. Microsoft lacks knowledge sufficient to form a belief as to the truth of the allegations in paragraph 2 of the Complaint, and on that basis denies such allegations.

3. Microsoft lacks knowledge sufficient to form a belief as to the truth of the allegations in paragraph 3 of the Complaint, and on that basis denies such allegations.

4. Microsoft lacks knowledge sufficient to form a belief as to the truth of the allegations in paragraph 4 of the Complaint, and on that basis denies such allegations.

5. Microsoft admits that it is a Washington corporation with its principal place of business at One Microsoft Way, Redmond, Washington 98052.

6. Microsoft admits that it distributes, markets, licenses, and offers to license the Windows 7 operating system throughout the United States, including in this district. Microsoft admits that it imports into the United States media containing the Windows 7 operating system. Microsoft denies that Internet Explorer 9 is an operating system for personal computers. Microsoft denies the allegations in paragraph 6 with respect to Internet Explorer 9, but admits that it makes available a beta version of Internet Explorer 9 throughout the United States, including in this district. Except as expressly admitted, Microsoft denies the allegations in paragraph 6 of the Complaint.

JURISDICTION AND VENUE

7. Microsoft admits that Motorola has alleged an action for patent infringement and that such actions arise under the patent laws of the United States, 35 U.S.C. §§ 101 *et seq*. However, Microsoft denies committing any infringement or other tortious or unlawful act. Microsoft admits that this Court has subject matter jurisdiction over claims arising under the patent laws of the United States pursuant to 28 U.S.C. §§ 1331 and 1338(a).

8. Microsoft admits that venue is proper in this Judicial District on the basis that Microsoft transacts business in this Judicial District, among others, but denies that venue is proper pursuant to 28 U.S.C. §§ 1391 (b), (c), and (d). Microsoft alleges that even if venue is proper, it is inconvenient. Microsoft denies that it has committed and/or induced any acts of patent infringement in this Judicial District or elsewhere. Further, Microsoft denies committing any tortious or unlawful act in this Judicial District or elsewhere.

9. Microsoft admits that it does business in this Judicial District and that this Court has personal jurisdiction over Microsoft for the purposes of this matter. Microsoft denies that it has committed acts of patent infringement or any tortious or unlawful acts in this District or elsewhere. Except as so admitted, Microsoft denies the allegations in paragraph 9 of the Complaint.

THE ASSERTED PATENTS

10. Microsoft admits that U.S. Patent No. 7,310,374 is entitled "Macroblock Level Adaptive Frame/Field Coding for Digital Video Content," bears an issuance date of December 18, 2007 and shows on its face that Limin Wang, Rajeev Gandhi, Krit Panusopone, and Ajay Luthra are the named inventors. Microsoft lacks knowledge sufficient to form a belief as to the

truth of the remaining allegations in paragraph 10 of the Complaint, and on that basis denies such allegations.

11. Microsoft admits that U.S. Patent No. 7,310,375 is entitled "Macroblock Level Adaptive Frame/Field Coding for Digital Video Content," bears an issuance date of December 18, 2007 and shows on its face that Limin Wang, Rajeev Gandhi, Krit Panusopone, and Ajay Luthra are the named inventors. Microsoft lacks knowledge sufficient to form a belief as to the truth of the remaining allegations in paragraph 11 of the Complaint, and on that basis denies such allegations.

12. Microsoft admits that U.S. Patent No. 7,310,376 is entitled "Macroblock Level Adaptive Frame/Field Coding for Digital Video Content," bears an issuance date of December 18, 2007 and shows on its face that Limin Wang, Rajeev Gandhi, Krit Panusopone, and Ajay Luthra are the named inventors. Microsoft lacks knowledge sufficient to form a belief as to the truth of the remaining allegations in paragraph 12 of the Complaint, and on that basis denies such allegations.

13. Microsoft admits that it has had knowledge of the '374, '375, and '376 patents since October 29, 2010. Microsoft admits that it has been involved in litigation with Motorola since October 1, 2010. Except as so admitted, Microsoft denies the allegations in paragraph 13 of the Complaint.

<u>CLAIM ONE</u>

([Alleged] Infringement of U.S. Patent No. 7,310,374)

14. Microsoft incorporates by reference paragraphs 1-13 above.

15. Microsoft denies the allegations in paragraph 15 of the Complaint.

16. Microsoft denies the allegations in paragraph 16 of the Complaint.

17. Microsoft denies the allegations in paragraph 17 of the Complaint.

18. Microsoft denies the allegations in paragraph 18 of the Complaint.

19. Microsoft denies that it has committed any acts of infringement. Microsoft further denies that Motorola has been damaged by Microsoft's alleged activities or that Motorola is entitled to any form of injunctive relief on account of said alleged acts.

20. Microsoft denies that it has ever infringed or is currently infringing the '374 Patent. Microsoft further denies that Motorola has been or continues to be damaged by Microsoft's alleged activities.

21. Microsoft denies that it has ever infringed or is currently infringing the '374 Patent, willfully or in any other way.

22. Microsoft denies the allegations in paragraph 22 of the Complaint.

<u>CLAIM TWO</u>

([Alleged] Infringement of U.S. Patent No. 7,310,375)

23. Microsoft incorporates by reference paragraphs 1-13 above.

24. Microsoft denies the allegations in paragraph 24 of the Complaint.

25. Microsoft denies the allegations in paragraph 25 of the Complaint.

26. Microsoft denies the allegations in paragraph 26 of the Complaint.

27. Microsoft denies the allegations in paragraph 27 of the Complaint.

28. Microsoft denies that it has committed any acts of infringement. Microsoft further denies that Motorola has been damaged by Microsoft's alleged activities or that Motorola is entitled to any form of injunctive relief on account of said alleged acts.

29. Microsoft denies that it has ever infringed or is currently infringing the '375 Patent. Microsoft further denies that Motorola has been or continues to be damaged by Microsoft's alleged activities.

30. Microsoft denies that it has ever infringed or is currently infringing the ’375 Patent, willfully or in any other way.

31. Microsoft denies the allegations in paragraph 31 of the Complaint.

CLAIM THREE

([Alleged] Infringement of U.S. Patent No. 7,310,376)

32. Microsoft incorporates by reference paragraphs 1-13 above.

33. Microsoft denies the allegations in paragraph 33 of the Complaint.

34. Microsoft denies the allegations in paragraph 34 of the Complaint.

35. Microsoft denies the allegations in paragraph 35 of the Complaint.

36. Microsoft denies the allegations in paragraph 36 of the Complaint.

37. Microsoft denies that it has committed any acts of infringement. Microsoft further denies that Motorola has been damaged by Microsoft’s alleged activities or that Motorola is entitled to any form of injunctive relief on account of said alleged acts.

38. Microsoft denies that it has ever infringed or is currently infringing the ’376 Patent. Microsoft further denies that Motorola has been or continues to be damaged by Microsoft’s alleged activities.

39. Microsoft denies that it has ever infringed or is currently infringing the ’376 Patent, willfully or in any other way.

40. Microsoft denies the allegations in paragraph 40 of the Complaint.

JURY DEMAND

Microsoft acknowledges and joins in Motorola’s demand for a trial by jury on all claims and all issues triable by jury in this action.

PRAYER FOR RELIEF

Microsoft denies that Motorola is entitled to any of the relief requested in its prayer for relief or any relief whatsoever.

Microsoft denies all allegations of the Complaint not specifically admitted above.

<u>AFFIRMATIVE DEFENSES</u>

First Affirmative Defense

41. On information and belief, Microsoft has not been and is not now infringing any valid and enforceable claim of the ’374, ’375, or ’376 Patents, either directly or indirectly, literally or under the doctrine of equivalents, willfully or otherwise.

Second Affirmative Defense

42. On information and belief, each and every claim of the ’374, ’375, and ’376 Patents is invalid for failure to comply with the conditions of patentability, including but not limited to 35 U.S.C. §§ 101, 102, 103, 111, 112, 113, and/or 133.

Third Affirmative Defense

43. On information and belief, Motorola has inexcusably delayed filing this suit for an unreasonable period of time to the material prejudice of Microsoft and is now barred from recovery of pre-suit damages because of laches.

Fourth Affirmative Defense

44. On information and belief, Motorola is estopped by representations or actions taken during the prosecution of the ’374, ’375, and ’376 Patents and related patents under the doctrine of prosecution history estoppel.

Fifth Affirmative Defense

45. On information and belief, the claims of the ’374, ’375, and ’376 Patents are barred by license, equitable estoppel and/or waiver.

Sixth Affirmative Defense

46. To the extent Motorola seeks damages for alleged infringement more than six years prior to the filing of the present litigation, Motorola's claims are barred by the statute of limitations under 35 U.S.C. § 286.

Seventh Affirmative Defense

47. On information and belief, Motorola's remedies are limited under 35 U.S.C. § 287.

Eighth Affirmative Defense

48. Motorola's demand to enjoin Microsoft is barred, as Motorola has suffered neither harm nor irreparable harm from Microsoft's actions.

Ninth Affirmative Defense

49. To the extent that Motorola's claims relate to the sale to and/or use by or for the United States government of the allegedly infringing products, Motorola's claims for relief are barred by 28 U.S.C. § 1498.

Tenth Affirmative Defense

50. As more fully described in Microsoft's currently pending Motion to Dismiss, Stay, or in the Alternative, Transfer Venue (Dkt. 25-26), Motorola's infringement claims are barred from being litigated in this action because they are compulsory counterclaims to Microsoft's first-filed action in the Western District of Washington (Case No. 10-1823).

Eleventh Affirmative Defense

51. Microsoft reserves all affirmative defenses under Rule 8(c) of the Federal Rules of Civil Procedure, the Patent Laws of the United States, and any other defenses, at law or in

equity, that may now exist or in the future be available based on discovery and further factual investigation in this case.

COUNTERCLAIMS

Microsoft Corporation ("Microsoft") alleges as follows against Motorola Mobility, Inc. and General Instrument Corporation (collectively "Motorola"):

1. Microsoft is a corporation organized and existing under the laws of the State of Washington, with its principal place of business at One Microsoft Way, Redmond, Washington 98052.

2. On information and belief, Defendant Motorola Mobility, Inc. is organized under the laws of Delaware having a principal place of business at 600 North U.S. Highway 45, Libertyville, Illinois 60048. On information and belief, Defendant General Instrument Corporation is a wholly-owned subsidiary of Motorola Mobility, Inc. and is organized under the laws of Delaware having a principal place of business at 101 Tournament Drive, Horsham, Pennsylvania 19044. Motorola Mobility, Inc., and General Instrument Corporation will be referred to collectively herein as "Motorola" or "Defendant".

JURISDICTION AND VENUE

3. Microsoft repeats and realleges the allegations of paragraphs 1 through 2 of these Counterclaims in their entirety.

4. Microsoft brings these counterclaims under the patent laws of the United States, 35 U.S.C. § 1, *et seq.* and the Declaratory Judgment Act, 28 U.S.C. §§ 2201-2202. Thus, this Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 & 1338(a).

5. Venue is proper in this district under 28 U.S.C. §§ 1391(a), (c), and (d) and 1400(b) and because Motorola is subject to personal jurisdiction in this district.

6. Personal jurisdiction is proper in this district at least because Motorola has consented to jurisdiction in this district by filing suit against Microsoft in this Court.

RELATED LITIGATION

7. On November 9, 2010, Microsoft sued Motorola in the Western District of Washington alleging breach of contract, promissory estoppels, and wavier (the "Washington action"). On November 10, 2010, Plaintiffs filed this suit. On December 21, 2010, Microsoft filed a motion to dismiss, stay, or transfer this case on the grounds that the infringement claims asserted by Plaintiffs are compulsory counterclaims to Microsoft's Washington action. (Dkt. 25-26.)

8. Microsoft files the counterclaims below as of right in response to Motorola's Complaint, but maintains that the patents asserted by Motorola in this case may only be properly litigated as counterclaims to the first-filed Washington action, thereby requiring the dismissal of this entire case. Microsoft has filed this answer and these counterclaims only because the Court has not yet ruled on Microsoft's pending motion to dismiss, stay, or transfer this case, and Microsoft will dismiss these counterclaims upon the dismissal of Motorola's affirmative claims.

THE MICROSOFT ASSERTED PATENTS

9. U.S. Patent No. 6,339,780 ("the '780 Patent"), entitled "Loading Status in a Hypermedia Browser Having a Limited Available Display Area," issued on January 15, 2002 and names Scott R. Shell, Kevin T. Shields, and Anthony Kitowitz as inventors. Microsoft is the owner of all right, title, and interest in and to the '780 Patent, including the right to sue and recover for past infringement thereof. A true and correct copy of the '780 Patent is attached as Exhibit A.

10. U.S. Patent No. 7,411,582 ("the '582 Patent"), entitled "Soft Input Panel System and Method," issued on August 12, 2008 and names Michael G. Toepke, Jeffrey R. Blum, and Kathryn L. Parker as inventors. Microsoft is the owner of all right, title, and interest in and to the '582 Patent, including the right to sue and recover for past infringement thereof. A true and correct copy of the '582 Patent is attached as Exhibit B.

MICROSOFT'S FIRST COUNT

(Motorola's Infringement of U.S. Patent No. 6,339,780)

11. Microsoft repeats and realleges the allegations of paragraphs 1 through 10 of these Counterclaims in their entirety.

12. On information and belief, Motorola has infringed, induced infringement of and/or contributorily infringed and continues to infringe, induce infringement of and/or contributorily infringe, at least independent claims 1, 12, 32, 36, and 40 of the '780 Patent, pursuant to 35 U.S.C. § 271(a), (b) and/or (c), literally or under the doctrine of equivalents, in this district and elsewhere in the United States, by making, using, selling, offering to sell and/or importing products such as Android smartphones including, *e.g.*, at least one or more of the following: the Motorola Droid X, the Motorola Droid 2, the Motorola Devour and the Motorola Charm.

13. On information and belief, instructional materials provided by Motorola (available at, *e.g.*, http://www.motorola.com/consumers/v/index.jsp?vgnextoid=25aae66506e9d110VgnVCM1000008406b00aRCRD, and http://www.motorola.com/Support/US-EN/Consumer-Support/Mobile-Phones/Motorola+DROID+X) instruct customers how to use these products in accordance with at least independent claims 1, 12, 32, 36, and 40 of the '780 Patent.

14. Microsoft is entitled to recover damages adequate to compensate it for Motorola's infringement, and in any event no less than a reasonable royalty.

15. Motorola's infringing activities have caused and will continue to cause Microsoft irreparable harm unless the infringement is enjoined by this Court.

MICROSOFT'S SECOND COUNT

(Motorola's Infringement of U.S. Patent No. 7,411,582)

16. Microsoft repeats and realleges the allegations of paragraphs 1 through 10 of these Counterclaims in their entirety.

17. On information and belief, Motorola has infringed, induced infringement of and/or contributorily infringed and continues to infringe, induce infringement of and/or contributorily infringe, at least independent claims 11 and 19 of the '582 Patent, pursuant to 35 U.S.C. § 271 (a), (b) and/or (c), literally or under the doctrine of equivalents, in this district and elsewhere in the United States, by making, using, selling, offering to sell and/or importing products such as Android smartphones including, *e.g.*, at least one or more of the following: the Motorola Droid X, the Motorola Droid 2, the Motorola Cliq XT, and the Motorola i1.

18. On information and belief, instructional materials provided by Motorola (available at, *e.g.*, http://www.motorola.com/consumers/v/index.jsp?vgnextoid=25aae66506e9d110VgnVCM1000008406b00aRCRD, and http://www.motorola.com/Support/US-EN/Consumer-Support/Mobile-Phones/Motorola+DROID+X) instruct customers how to use these products in accordance with at least independent claims 11 and 19 of the '582 Patent.

19. Microsoft is entitled to recover damages adequate to compensate it for Motorola's infringement, and in any event no less than a reasonable royalty.

20. Motorola's infringing activities have caused and will continue to cause Microsoft irreparable harm unless the infringement is enjoined by this Court.

MICROSOFT'S THIRD COUNT

(Breach of RAND Licensing Obligations)

21. On November 9, 2010, Microsoft sued Motorola in the Western District of Washington alleging breach of contract, promissory estoppel, and waiver (the "Washington action"). On November 10, 2010, Plaintiffs filed this suit. On December 21, 2010, Microsoft filed a motion to dismiss, stay, or transfer this case on the grounds that the infringement claims asserted by Plaintiffs are compulsory counterclaims to Microsoft's Washington action. (Dkt. 25-26.) Subject to resolution of Microsoft's pending motion, Microsoft conditionally re-alleges and asserts that Motorola's relief for any infringement of the patents in suit shall be no more than "RAND" terms consistent with its obligations and representations to the relevant Standard Developing Organization, for the reasons set forth in Microsoft's forthcoming Amended Complaint in the Washington action . The substance of the Amended Complaint in the Washington action is set forth below.

MICROSOFT'S THIRD COUNT – NATURE OF THE ACTION

22. Microsoft brings this action for Motorola's breach of its commitments to the Institute of Electrical and Electronics Engineers Standards Association ("IEEE-SA"), International Telecommunications Union ("ITU"), and their members and affiliates – including Microsoft. Motorola broke its promises to license patents it asserted as related to wireless technologies known as "WLAN" and to video coding technologies generally known as "H.264" under reasonable rates, with reasonable terms, and under non-discriminatory conditions.

23. Participants in IEEE-SA standards setting efforts, including those directed to

WLAN technology, were subject to the IEEE-SA Standard Board Bylaws concerning the submission of Letters of Assurance related to patent claims deemed "essential" by a submitting party. Clause 6 of those Bylaws (which was revised slightly over the years) generally provides in pertinent part:

> A Letter of Assurance shall be either:
>
> a) A general disclaimer to the effect that the submitter without conditions will not enforce any present or future Essential Patent Claims against any person or entity making, using, selling, offering to sell, importing, distributing, or implementing a compliant implementation of the standard; or
>
> b) A statement that a license for a compliant implementation of the standard will be made available to an unrestricted number of applicants on a worldwide basis without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination.

24. Motorola openly and publicly submitted Letters of Assurance pursuant to Clause 6 of the IEEE-SA Standards Board Bylaws that it would offer to license any of its patents that it identified as "essential" to the applicable WLAN standard(s) to any entity under reasonable rates on a non-discriminatory basis. IEEE-SA and its participants and affiliates relied on Motorola's promises in developing, adopting and implementing IEEE-SA technical standards. These standards are now implemented worldwide in a variety of electronic devices that have become commonplace. Microsoft invested substantial resources in developing and marketing products in compliance with these standards, relying on the assurances of participating patent holders – including Motorola – that any patents asserted to be "essential" by such patent holders would be available for licensing on such terms, regardless of whether such patents were, in fact, used in any particular implementation.

25. Participants in ITU standards setting efforts, including those directed to H.264 technology, were subject to the ITU-T Common Patent Policy concerning the submission of Patent Statement and Licensing Declaration related to patents identified by a submitting party.

ITU-T Common Patent Policy generally provides, in pertinent part, that a patent holder's statement may declare that:

> (2.1) The patent holder is willing to negotiate licenses free of charge with other parties on a non-discriminatory basis on reasonable terms and conditions.
>
> (2.2) The patent holder is willing to negotiate licenses with other parties on a non-discriminatory basis on reasonable terms and conditions.

26. Motorola openly and publicly submitted Patent Statement and Licensing Declarations pursuant to the ITU's Common Patent Policy that it would offer to license any of its patents that it identified for the H.264 technologies to any entity under reasonable rates on a non-discriminatory basis. The ITU and its participants and affiliates relied on Motorola's promises in developing, adopting and implementing ITU H.264 technical standards. These standards are now implemented worldwide in a variety of electronic devices and software that have become commonplace. Microsoft invested substantial resources in developing and marketing products in compliance with these standards, relying on the assurances of participating patent holders – including Motorola – that any patents identified pursuant to ITU's Common Patent Policy by such patent holders would be available for licensing on such terms, regardless of whether such patents were, in fact, used in any particular implementation.

27. Motorola broke its promise to IEEE-SA and its members and affiliates by refusing to offer to Microsoft a license that is consistent with Clause 6 of IEEE-SA Standards Board Bylaws, instead demanding royalties that are excessive and discriminatory. Motorola broke its promise to ITU and its members and affiliates by refusing to offer to Microsoft a license that is consistent with the Common Patent Policy of the ITU, instead demanding royalties that are excessive and discriminatory.

28. Microsoft does not accept Motorola's representation that any of its patents that it has identified to the IEEE or ITU are, in fact, necessary to the implementation of compliant

implementations of WLAN or H.264 technologies; nor does Microsoft concede that the particular implementations of such technologies in its products practice any Motorola patents, including those identified by Motorola in relation to these technologies. Nonetheless, Microsoft has relied upon Motorola's, and other similarly-situated patent holders', representations that all patent controversies may be avoided based on the offer of patent licenses on reasonable rates and non-discriminatory terms.

29. Motorola's breach of its commitments does not depend on whether any Motorola patents which Motorola has identified in relation to standards are, in fact, "essential" to practicing those standards, whether those standards can be practiced in ways that do not infringe the identified Motorola patents or whether Microsoft has infringed any valid Motorola patents. Because Motorola promised that it would license any such patents on reasonable and non-discriminatory terms, companies that rely on those commitments are entitled to avoid becoming embroiled in patent controversies and to receive the benefit of an offer of a reasonable and non-discriminatory license.

30. Accordingly, Microsoft seeks: i) a judicial declaration that Motorola's promises to IEEE-SA, the ITU, and their respective members and affiliates constitute contractual obligations that are binding and enforceable by Microsoft; ii) a judicial declaration that Motorola has breached these obligations by demanding excessive and discriminatory royalties from Microsoft; iii) a judicial accounting of what constitutes a royalty rate in all respects consistent with Motorola's promises for WLAN patents identified as "essential" by Motorola and for H.264 patents identified by Motorola; and iv) a judicial determination of and compensation for Motorola's breach.

<u>MICROSOFT'S THIRD COUNT – PARTIES</u>

31. Counterclaim Plaintiff Microsoft is a Washington corporation having its principal place of business at One Microsoft Way, Redmond, Washington 98052.

32. Founded in 1975, Microsoft is a worldwide leader in computer software, services, and solutions for businesses and consumers. Since 1979, Microsoft has been headquartered in the Redmond, Washington area. Microsoft currently employs nearly 40,000 people in the Puget Sound region and occupies nearly 8 million square feet of facilities at its Redmond campus.

33. Microsoft has a long history of technical innovation in the software and hardware products it develops and distributes.

34. Microsoft's products include Xbox video game consoles, various versions of which have been sold to consumers since 2001. Xbox has grown in popularity over the years and is now one of the most widely-sold video game consoles on the market.

35. Over the years that Xbox has been sold, some versions have had wireless Internet connectivity ("WLAN") built-in and some versions have had optional WLAN connectivity. All versions of Xbox that include hardware and software that allows for WLAN connectivity also offer an alternative, wired connection to the Internet. Xbox video game consoles function as video game consoles, regardless of their ability to connect to the Internet.

36. Microsoft relies upon third-party suppliers to provide an interface to WLAN connections. The WLAN interface provided by these third-parties is one of many components that underlie the operation and functionality of the Xbox consoles. The WLAN interface does not enable any of Xbox's core video gaming functionality. Instead, it simply enables WLAN connectivity for those consumers who choose to use that functionality.

37. Microsoft hardware and software products that provide users with H.264 technologies further provide substantial other features and functions. By way of non-limiting

example, personal computers in various configurations offer the end-user myriad features and functionality. H.264 technologies provided through Microsoft software supplied to computer and other equipment makers represent but a fraction of the end price for such products. By way of further non-limiting example, Microsoft's Xbox video game console provides video game play without reliance upon any H.264 technologies that may be made available to users through other features and functions.

38. Microsoft also relies upon third-party suppliers in at least some instances for H.264 technologies.

39. On information and belief, Defendant Motorola Mobility, Inc. is organized under the laws of Delaware having a principal place of business at 600 North U.S. Highway 45, Libertyville, Illinois 60048. On information and belief, Defendant General Instrument Corporation is a wholly-owned subsidiary of Motorola Mobility, Inc. and is organized under the laws of Delaware having a principal place of business at 101 Tournament Drive, Horsham, Pennsylvania 19044. Motorola Mobility, Inc., and General Instrument Corporation will be referred to collectively herein as "Motorola" or "Defendant".

MICROSOFT'S THIRD COUNT – JURISDICTION AND VENUE

40. This Court has jurisdiction over the subject matter of this dispute pursuant to 28 U.S.C. § 1332, because this is an action between citizens of different states and because the value of declaratory and injunctive relief sought, the value of Microsoft's rights this action will protect and enforce, Microsoft's damages, and the extent of the injury to be prevented exceed the amount of $75,000, exclusive of interest and costs.

41. On information and belief, Defendant is subject to this Court's personal jurisdiction, consistent with the principles of due process and the Washington Long Arm Statute,

at least because Defendant maintains offices and facilities in the Western District of Washington, offers its products for sale in the Western District of Washington, and/or has transacted business in this District.

42. Venue is proper in this district pursuant to 28 U.S.C. §§ 1391(a), 1391(c), and 1391(d).

MICROSOFT'S THIRD COUNT – BACKGROUND

Introduction to Standards

43. New wireless and video coding technologies typically are only broadly commercialized after service providers and device manufacturers agree on compatible technology specifications for related products or services. For virtually all successful wireless and video coding technologies, that process has involved inclusive, multi-participant standards development efforts conducted under the auspices of leading standards development organizations.

44. Standards play a critical role in the development of wireless and video coding technologies. Standards facilitate the adoption and advancement of technology as well as the development of products that can interoperate with one another. Companies that produce products compatible with a standard can design products by referencing only the standard documentation, without the need to communicate separately with every other company with which their products may need to interoperate. Companies producing products that implement and are tested to a standard can therefore be confident that their products will operate with other products that also are compatible with that standard, and consumers of those products can be confident that products from multiple vendors will work together as intended under the standard.

45. As a practical matter, the technologies that are used to allow a consumer

electronics device to connect wirelessly to the Internet must be described in standards adopted by a recognized SDO (standard development organization), and thereby accepted by key industry members, in order to be commercially successful. For example, Microsoft could not purchase third-party goods that enable its Xbox devices to connect wirelessly to the Internet unless those goods were compatible with standards described by an SDO.

46. Correspondingly, video technologies that are used to allow a consumer electronics device to display video encoded pursuant to any particular coding protocol must be described in standards adopted by a recognized SDO, and thereby accepted by key industry members, in order to be commercially successful. For example, Microsoft and computer makers could not purchase third-party products or software that provide reliable video decoding and image generation unless those products or software were compatible with standards described by an SDO.

47. In order to reduce the likelihood that implementers of their standards will be subject to abusive practices by patent holders, SDOs have adopted rules, policies and procedures that address the disclosure and licensing of patents that SDO participants may assert in relation to the practice of the standard under consideration. These rules, policies and/or procedures are set out in the intellectual property rights policies ("IPR policies") of the SDOs.

48. Many IPR policies – including those at issue in this litigation – encourage or require participants to disclose on a timely basis the IPR, such as patents or patent applications, that they believe are sufficiently relevant to standards under consideration. These disclosures permit the SDOs and their members to evaluate technologies with full knowledge of disclosed IPR that may affect the costs of implementing the standard.

49. IPR policies – including those at issue in this litigation – require participants claiming to own relevant patents to negotiate licenses for those patents with any implementer of

the standard on reasonable and non-discriminatory terms. As their inclusion in the IPR policies of various standards development organizations suggests, such commitments are crucial to the standards development process. They enable participants in standards development to craft technology standards with the expectation that an owner of any patented technology will be prevented from demanding unfair, unreasonable, or discriminatory licensing terms and thereby be prevented from keeping parties seeking to implement the standard from doing so or imposing undue costs or burdens on them.

Wireless LAN Standards

50. Motorola's unlawful licensing demands pertain in part to patents that it claims are "essential" to a widely practiced standard for wireless Internet connectivity known as "WLAN," "Wi-Fi," and/or "802.11."

51. WLAN enables an electronic device to access the Internet wirelessly at high speeds over short distances. WLAN networks typically consist of one or more access points that are connected to an Ethernet local area network, each of which communicates by radio signals with devices such as notebook computers and other electronics devices.

52. The use of WLAN technology has grown in the United States since its introduction in the 1990s. Manufacturers now offer WLAN connectivity in various devices for various reasons.

53. WLAN is based on the 802.11 wireless networking standard developed by the Institute of Electrical and Electronics Engineers ("IEEE") beginning in the early 1990s. The initial 802.11 protocol ("legacy 802.11") was released in 1997. Since then, there have been a number of amendments issued, the most important of which are 802.11a (1999), 802.11b (1999), 802.11g (2003), and 802.11n (2009).

<u>**H.264 Standards**</u>

54. Motorola's unlawful licensing demands pertain in part to patents that it has identified to the ITU and its members in relation to H.264 technologies.

55. H.264 technologies provide video decoding in such applications as DVD players, videos available for downloading or replay on the Internet, web software, broadcast services, direct-broadcast satellite television services, cable television services, and real-time videoconferencing.

56. The use of H.264 technology has grown in the United States since its introduction. Manufacturers now offer H.264 connectivity in various software and devices for various reasons.

57. H.264 technology was developed as a standard set of technologies at least in part through the auspices of the International Telecommunications Union ("ITU").

<u>**Motorola's Involvement in Development of the WLAN Standards**</u>

58. The standard setting arm of IEEE, the IEEE Standards Association ("IEEE-SA"), promulgates technical standards in a variety of fields, including telecommunications. IEEE-SA had an IPR policy at the time it was drafting the 802.11 (WLAN) protocols. Under the IPR policy, when individuals participating in IEEE standards development came to believe that a company, university, or other patent holder owned patents or patent applications that might be "essential" to implement an IEEE standard under development, IEEE-SA would request Letters of Assurance from those entities.

59. The requirements for the Letters of Assurance sought by IEEE are set forth in Clause 6 of the IEEE-SA Standards Board Bylaws.

60. According to IEEE's IPR policy, Letters of Assurance, once provided, are

irrevocable and shall be in force at least until the standard's withdrawal.

61. If the Letters of Assurance were not provided for patents asserted to be "essential" by participants, the IEEE working group either would revise the standard so that compliance could be achieved without facing any potential issues related to such patent(s), discontinue work on the standard altogether, or otherwise proceed in a manner consistent with the non-disclosure and lack of Letters of Assurance so that participating and relying entities would not be exposed to discriminatory patent assertions and/or unreasonable licensing terms.

62. Motorola has represented to Microsoft that it owns rights in a number of patents and pending applications that it asserts are or may become "essential" to comply with one or more amendments to the 802.11 standard. By way of example, Motorola has represented to Microsoft that the following patents, among others, are or may become "essential" to comply with one or more amendments to the 802.11 standard: U.S. Patent Nos. 5,319,712; 5,311,516; 5,572,193; 5,311,516; and 5,636,223. The full list of patents is provided in Exhibit C. Microsoft does not concede that such listed patents are either "essential" to the 802.11 standards or that such patents are practiced in the implementation of such standards in any Microsoft products.

63. On information and belief, Motorola obtained rights to several of THE WLAN patents it has represented as "essential" through its recent acquisition of Symbol Technologies, Inc. ("Symbol").

64. Prior to the releases of the 802.11 protocols, Motorola and Symbol submitted Letters of Assurance to the IEEE pursuant to Clause 6 of the IEEE-SA Standards Board Bylaws with respect to those protocols, guaranteeing that any "essential" patents would be licensed under reasonable and non-discriminatory terms and conditions. Both Motorola's and Symbol's Letters

of Assurance apply to any "essential" patents they then held as well as any other "essential" patents they subsequently obtained.

65. In reliance on these letters of assurance, IEEE released the 802.11 standard and various amendments to that standard which Motorola asserts incorporated Motorola's and Symbol's patented technology. On information and belief, absent the Letters of Assurance, the relevant IEEE working groups would have either revised the standards, employing alternative technologies instead, or stopped working on the protocols.

66. In submitting its Letter of Assurance pursuant to the applicable IEEE IPR policy, Motorola entered into an actual or implied contract with IEEE, for the benefit of IEEE members and any entity that implements the 802.11 standard. Motorola is bound by its agreements to offer licenses consistent with the referenced IEEE bylaws.

67. Similarly, Symbol, in submitting its Letter of Assurance pursuant to the applicable IEEE IPR policy, entered into an actual or implied contract with IEEE, for the benefit of IEEE members and any other entity that implements the 802.11 standard, and Motorola is bound by that commitment.

Motorola's Involvement in Development of the H.264 Standards

68. ITU is the leading United Nations agency for information and communication technology issues, and the global focal point for governments and the private sector in developing networks and services. ITU historically has coordinated the shared global use of the radio spectrum, promoted international cooperation in assigning satellite orbits, worked to improve telecommunication infrastructure in the developing world, established the worldwide standards that foster seamless interconnection of a vast range of communications systems and addressed the global challenges of our times, such as strengthening cybersecurity.

69. In conjunction with its efforts to provide standards in support of its stated goals, the ITU requires that its members and participants adhere to the Common Patent Policy stated above.

70. According to ITU's IPR policy, Patent Statement and Licensing Declarations, once provided, are irrevocable and shall be in force at least until the standard's withdrawal.

71. If the Patent Statement and Licensing Declarations were not provided for relevant patents from participants, the ITU either would revise the standard so that compliance could be achieved without facing any potential issues related to such patent(s), discontinue work on the standard altogether, or otherwise proceed in a manner consistent with the non-disclosure and lack of Patent Statement and Licensing Declarations so that participating and relying entities would not be exposed to discriminatory patent assertions and/or unreasonable licensing terms.

72. Motorola has represented to Microsoft and others that it owns rights in a number of patents and pending applications that are or may be embodied fully or partly within H.264 technologies as endorsed by ITU and has identified these patents to the ITU. Microsoft does not concede that such listed patents are either "essential" to the 802.11 standards or that such patents are practiced in the implementation of such standards in any Microsoft products.

73. Motorola submitted Patent Statement and Licensing Declarations to the ITU pursuant to its Common Patent Policy with respect to those protocols, guaranteeing that Motorola's identified patents would be licensed under reasonable and non-discriminatory terms and conditions.

74. In reliance on these Patent Statement and Licensing Declarations, ITU proceeded with the H.264 standard and various amendments to that standard which Motorola asserts incorporated Motorola's patented technology. On information and belief, absent the Patent

Statement and Licensing Declarations, the ITU would have either revised the standards, employing alternative technologies instead, or stopped working on the protocols.

75. In submitting its Patent Statement and Licensing Declarations pursuant to the applicable ITU policy, Motorola entered into an actual or implied contract with ITU, for the benefit of ITU members and any entity that implements the H.264 technologies. Motorola is bound by its agreements to offer licenses consistent with the referenced ITU Common Patent Policy.

<u>Microsoft's Reliance on Commitments with Respect to WLAN and H.264 Technologies</u>

76. Microsoft has participated in the development of the IEEE WLAN standards.

77. Microsoft and other companies participating in the development of WLAN in IEEE relied on Motorola's commitments to ensure that the royalties Motorola would seek would conform to the promises made by Motorola.

78. In reliance on the integrity of the SDO process and the commitments made by Motorola and others regarding WLAN patents they deem "essential," Microsoft began providing its Xbox video game consoles with WLAN connectivity. By way of example, Microsoft purchased and incorporated into its Xbox 360 video game consoles third-party-manufactured interfaces that provide Xbox 360 devices with WLAN connectivity. Microsoft made its decision to provide its Xbox video game consoles with WLAN connectivity in reliance on, and under the assumption that, it and/or any third party supplier could avoid patent litigation and take a license to any patents that Motorola, or any other company, has disclosed to the WLAN standard under IEEE's well publicized IPR policy.

79. Microsoft and other manufacturers of WLAN-compliant devices necessarily relied on the commitments of Motorola and others to disclose and license any identified patents

under these terms to avoid any patent controversy even if such patents are not necessary to compliant implementations nor actually practiced in any particular implementation.

80. Microsoft has participated in the development of the H.264 technologies.

81. Microsoft and other companies participating in the development of H.264 under the auspices of the ITU relied on Motorola's commitments to ensure that the royalties Motorola would seek for identified patents would conform to the promises made by Motorola.

82. Correspondingly, in reliance on the integrity of the SDO process and specifically the commitments made by Motorola and others regarding patents related to H.264 technologies, Microsoft began providing its H.264 technology capability in its Xbox video game consoles. Microsoft made its decision to provide its Xbox video game consoles with H.264 technology in reliance on, and under the assumption that, it and/or any third party supplier could avoid patent litigation and take a license to any patents that Motorola, or any other company, has disclosed to the ITU under its well-publicized IPR policy.

83. Microsoft made similar investments in other fields, including Windows 7 and Windows Phone 7, based upon Motorola's representations in relation to the H.264 technology standards.

84. Microsoft and other manufacturers and suppliers of H.264 compliant technology necessarily relied on the commitments of Motorola and others to license their identified patents under these terms to avoid any patent controversy even if such patents are not necessary to compliant implementations nor actually practiced in any particular implementation.

Motorola's Breach of Its Contractual Obligation to License Its Identified Patents on The Promised Terms

85. In willful disregard of the commitments it made to IEEE and the ITU, Motorola

has refused to extend to Microsoft a license consistent with Motorola's promises for any of Motorola's identified patents.

86. Instead, Motorola is demanding royalty payments that are wholly disproportionate to the royalty rate that its patents should command under any reasonable calculus. Motorola has discriminatorily chosen Microsoft's Xbox product line and other multi-function, many-featured products and software, such as Windows 7 and Windows Phone 7 and products incorporating Microsoft software, for the purpose of extracting unreasonable royalties from Microsoft.

87. By way of non-limiting example, each Xbox device includes substantial software and many computer chips and modules that perform various functions, including to enable Xbox's core functionality as a video gaming machine. Of those, the Xbox console includes one – an interface provided to Microsoft by third-parties – that allows consumers optionally to connect an Xbox to the Internet using a WLAN connection.

88. The third-party WLAN interface does not enable any of Xbox's core video gaming functionality. In addition, Microsoft allows consumers an alternative, wired method to connect to the Internet. This alternative method does not require use of any WLAN technology.

89. By way of further non-limiting example, each personal computer running Windows 7 includes substantial software and many computer chips and modules that perform various functions, including those related to the general operation of a computing device. Of those, each personal computer includes just a portion directed to H.264 technologies.

90. By way of further non-limiting example, each smartphone running Windows Phone 7 includes substantial software and many computer chips and modules that perform various functions, including those related to the general and particularized operation of a smartphone independent of H.264 technology. Of those, each smartphone includes just a portion

directed to H.264 technologies.

91. By letter to Microsoft, dated October 21, 2010, Kirk Dailey, Motorola's Corporate Vice President Intellectual Property, stated that a royalty for a license to its purported "essential" patents must be based on "the price of the end product (e.g., each Xbox 360 product) and not on component software." The cost of the chips and associated components that provide wireless connectivity for Xbox 360 consoles is a small fraction of the overall cost of the device. Motorola thus seeks a royalty on components of Xbox 360 which are disproportionate to the value and contribution of its purportedly "essential" patents and has declined to offer a license to its purported "essential" patents unless it receives exorbitant and discriminatory royalty payments to which it is not entitled. On information and belief, Motorola has not previously entered into a license agreement for its purported "essential" patents that is comparable to the demand made of Microsoft. Motorola has thereby refused to offer to license the patents at a reasonable rate, with reasonable terms, under conditions that are demonstrably free of any unfair discrimination.

92. By letter to Microsoft, dated October 29, 2010, Kirk Dailey, Motorola's Corporate Vice President Intellectual Property, stated that a royalty for a license to its identified patents must be based on "the price of the end product (e.g., each Xbox 360 product, each PC/laptop, each smartphone, etc.) and not on component software (e.g., Xbox 360 system software, Windows 7 software, Windows Phone 7 software, etc.)." The cost such component software and any inter-related hardware is a small fraction of the overall cost of the listed devices. Motorola thus seeks a royalty on software and hardware components of Xbox 360 and other devices which are unrelated to its identified patents and has declined to offer a license unless it receives exorbitant royalty payments to which it is not entitled. On information and

belief, Motorola has not previously entered into a license agreement for its identified patents that is comparable to the demand made of Microsoft. Motorola has thereby refused to offer to license the patents at a reasonable rate, with reasonable terms, on a non-discriminatory basis.

93. Regardless of whether there exists any actual use of Motorola patent claims in any specific implementation that is compliant with the applicable standards, Motorola has represented that it possesses patents relevant to such implementations. On that basis, Motorola is required to tender an offer to license its identified patents in all respects consistent with its binding assurances to the IEEE, the ITU, and participating members.

94. Motorola's demands constitute a breach of its WLAN and H.264 commitments.

Motorola Files Several Patent Infringement Actions in Violation of its Contractual Commitments

95. On November 10, 2010, Motorola Mobility and General Instrument filed two complaints for patent infringement against Microsoft in the Federal District Court for the Western District of Wisconsin, Case No. 3:10-CV-699 (the "699 Action") and Case No. 3:10-CV-700 (the "700 Action")).

96. The 699 Action involves the following three patents: U.S. Patent Nos. 7,310,374; 7,310,375; and 7,310,376. These three patents are among those that Motorola claims are necessary or essential to practice the H.264 standard. In the 699 Action, Motorola is seeking – among other forms of relief – to permanently enjoin Microsoft from practicing these patents.

97. The 700 Action involves seven other patents: U.S. Patent Nos. 6,980,596; 7,162,094; 5,319,712; 5,357,571; 6,069,896; 5,311,516; and 6,686,931. At least six of these patents are among those that Motorola claims are necessary or essential to practice the WLAN or

H.264 standard. In the 700 Action, Motorola is seeking – among other forms of relief – to permanently enjoin Microsoft from practicing these patents.

98. On November 22, 2010, Motorola Mobility and General Instrument filed a complaint for patent infringement against Microsoft with the International Trade Commission ("ITC") captioned *In the Matter of Certain Gaming and Entertainment Consoles, Related Software, and Components Thereof* (ITC Case No. 337-TA-752) (the "ITC Action").

99. Motorola's ITC Action involves five patents: U.S. Patent Nos. 6,980,596; 7,162,094; 5,319,712; 5,357,571; and 6,069,896. All of these patents are among those that Motorola claims are necessary or essential to practice the WLAN or H.264 standard. In the ITC case, Motorola is seeking – among other forms of relief – to exclude Microsoft from importing, marketing, advertising, distributing, offering for sale, selling, or transferring any products that practice these patents.

100. The 699 Action, 700 Action, and Motorola's ITC Action are collectively referred to as the "Motorola Patent Actions." The patents that are the subject of the Motorola Patent Actions and that are also included among those patents that Motorola claims are necessary or essential to practice the WLAN or H.264 standard are hereafter referred to collectively as the "SDO Patents in Suit."

101. With respect to each of the SDO Patents in Suit, Motorola has refused to offer Microsoft a license consistent with Motorola's contractual undertakings to the IEEE-SA, ITU, and their participating members. Instead, Motorola has demanded royalty payments that are wholly disproportionate to the royalty rate that its patents should command under any reasonable calculus.

<u>MICROSOFT'S THIRD COUNT - CAUSE OF ACTION 3A</u>

(Breach Of Contract)

102. Microsoft realleges and incorporates by reference the allegations set forth in paragraphs 31-101 above.

103. Motorola entered into express or implied contractual commitments with IEEE-SA, the ITU and their respective members and affiliates relating to the WLAN standard and H.264 technologies.

104. Each third party that would potentially implement WLAN and H.264 technologies was an intended beneficiary of those contracts.

105. Motorola was contractually obligated to offer a license to its identified patents consistent with the applicable patent policy of the IEEE-SA Standards Board Bylaws and the ITU, respectively.

106. Motorola breached these contracts by refusing to offer licenses to its identified patents (including the SDO Patents in Suit) under reasonable rates, with reasonable terms, and on a non-discriminatory basis.

107. Motorola further breached these contracts by filing the Motorola Patent Actions, which seek to enjoin Microsoft's implementation of the technology of the SDO Patents in Suit and to exclude Microsoft from, among other things, importing or selling products that implement the technology of the SDO Patents in Suit. To the extent this technology is actually necessary to implementation of the relevant standards (as Motorola has asserted), Motorola was obligated to offer, and eventually to negotiate, licenses to Microsoft on RAND terms. Because of its SDO contractual duties and the benefits Motorola receives from inclusion of its technology in SDO standards, Motorola has no right to enjoin or exclude Microsoft from implementing the technology of the SDO Litigated Patents. Motorola has failed and refused to offer the SDO

Patents in Suit on RAND terms, and has initiated the Motorola Patent Actions seeking improperly to enjoin or exclude Microsoft from using the technology of the SDO Patents in Suit.

108. As a result of these contractual breaches, Microsoft has been injured in its business or property, including damages associated with the cost of defending the improperly filed Motorola Patent Actions, and is otherwise threatened by imminent loss of profits, loss of customers and potential customers, and loss of goodwill and product image.

109. Microsoft has suffered damages and irreparable harm, and will suffer further damage and irreparable harm, by reason of each and all of the acts, practices, breaches and conduct of Motorola alleged above until and unless the Court enjoins such acts, practices, and conduct.

<u>MICROSOFT'S THIRD COUNT - CAUSE OF ACTION 3B</u>

(Promissory Estoppel)

110. Microsoft realleges and incorporates by reference the allegations set forth in paragraphs 31-101 above.

111. Motorola made a clear and definite promise to potential licensees through its commitments to IEEE and the ITU that it would license identified patents under reasonable rates, with reasonable terms, and on a non-discriminatory basis.

112. The intended purpose of Motorola's promises was to induce reliance. Motorola knew or should have reasonably expected that this promise would induce companies producing products in wireless networking and H.264 technologies, like Microsoft, to develop products compliant with the relevant standards.

113. Microsoft developed and marketed its products and services in reliance on Motorola's promises, as described above, including making its products and services compliant

with WLAN technical standards and including H.264 technologies in various Microsoft product offerings.

114. Motorola is estopped from reneging on these promises to the IEEE and the ITU under the doctrine of promissory estoppel.

115. Microsoft has been harmed as a result of its reasonable reliance on Motorola's promises and is threatened by the imminent loss of profits, loss of customers and potential customers, and loss of goodwill and product image.

116. Microsoft will suffer irreparable injury by reason of the acts and conduct of Motorola alleged above until and unless the court enjoins such acts, practices and conduct.

<u>MICROSOFT'S THIRD COUNT - CAUSE OF ACTION 3C</u>

(Waiver)

117. Microsoft realleges and incorporates by reference the allegations set forth in paragraphs 31-101 above.

118. Motorola expressly stated in its declarations to IEEE and the ITU that it would license its identified patents under reasonable rates and non-discriminatory terms.

119. Through this express statement, Motorola voluntarily and intentionally waived its rights to obtain compensation for its identified patents for the WLAN and H.264 standards other than at reasonable rates and on non-discriminatory terms.

120. Microsoft will suffer irreparable injury by reason of the acts and conduct of Motorola alleged above until and unless the court enjoins such acts, practices, and conduct.

<u>MICROSOFT'S THIRD COUNT - CAUSE OF ACTION 3D</u>

(Declaratory Judgment That Motorola's Offers Do Not Comply with Its Obligations)

121. Microsoft realleges and incorporates by reference the allegations set forth in

paragraphs 31-101 above.

122. There is a dispute between the parties concerning whether Motorola has offered to license to Microsoft patents consistent with Motorola's declarations and the referenced policy of the IEEE-SA Standards Board and the ITU.

123. The dispute is of sufficient immediacy and reality to warrant the issuance of a declaratory judgment.

124. Microsoft is entitled to a declaratory judgment that Motorola has not offered license terms to Microsoft conforming to applicable legal requirements.

<u>PRAYER FOR RELIEF – MICROSOFT'S COUNT 1 AND 2</u>

WHEREFORE, Microsoft prays for a judgment in its favor and against Motorola:

A. That Motorola take nothing by way of its Complaint;

B. That Motorola's infringement claims are compulsory counterclaims to Microsoft's first-filed Western District of Washington action (Case No. 10-1823).

C. That Microsoft has not been and is not now infringing, contributorily infringing, or inducing infringement of any valid and enforceable claim of the '374, '375, and '376, literally or under the doctrine of equivalents, willfully or otherwise;

D. That the '374, '375, and '376 Patents are invalid;

E. That the Microsoft Asserted Patents are valid and enforceable;

F. That Motorola has infringed the Microsoft Asserted Patents;

G. That Motorola, and all persons acting in privity or concert with, or otherwise controlled by Motorola, be permanently enjoined from continued infringement of the Microsoft Asserted Patents;

H. That Microsoft be awarded damages for Motorola's infringement of the Microsoft Asserted Patents, including pre-judgment and post-judgment interest; and

I. That Microsoft be awarded its expenses, costs, and attorney's fees under 35 U.S.C. § 285, along with any other and further relief as the Court deems just and proper.

<u>PRAYER FOR RELIEF – MICROSOFT'S COUNT 3A-3D</u>

WHEREFORE, Microsoft prays for relief as follows:

J. Adjudge and decree that Motorola is liable for breach of contract;

K. Adjudge and decree that Motorola is liable for promissory estoppel;

L. Enter judgment against Motorola for the amount of damages that Microsoft proves at trial;

M. Enter a judgment awarding Microsoft its expenses, costs, and attorneys fees in accordance with Rule 54(d) of the Federal Rules of Civil Procedure;

N. Enjoin Motorola from further demanding excessive royalties from Microsoft that are not consistent with Motorola's obligations, and from enforcing, or seeking to enforce, patent infringement claims in the Motorola Patent Actions (or elsewhere) in breach of its RAND obligations as alleged above;

O. Decree that Motorola has not offered royalties to Microsoft under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination;

P. Decree that Microsoft is entitled to license from Motorola any and all patents that Motorola deems "essential" to WLAN technology under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination;

Q. Decree that Microsoft is entitled to license from Motorola any and all patents that Motorola has identified to the ITU in relation to H.264 technology on a non-discriminatory basis on reasonable terms and conditions; and

R. For such other and further relief as the Court deems just and proper.

DATED this 25th day of January 2011.

MICHAEL BEST & FRIEDRICH LLP

By: /s/ J. Donald Best
J. Donald Best, SBN 1012450
John C. Scheller, SBN 1031247
Christopher C. Davis, SBN 1064764
P.O Box 1806
Madison, Wisconsin 53701-1806
Tel: (608) 257-3501
Fax: (608) 283-2275
Email: *jdbest@michaelbest.com*
jcscheller@michaelbest.com
ccdavis@michaelbest.com

David T. Pritikin
dpritikin@sidley.com
Richard A. Cederoth
rcederoth@sidley.com
John W. McBride
jwmcbride@sidley.com
SIDLEY AUSTIN LLP
One South Dearborn
Chicago, Illinois 60603
Tel. (312) 853-7000

Attorneys for Defendant
Microsoft Corporation

Of Counsel:

Douglas I. Lewis *(pro hac vice to be filed)*
dilewis@sidley.com
SIDLEY AUSTIN LLP
One South Dearborn

Chicago, Illinois 60603
Tel. (312) 853-7000

T. Andrew Culbert
andycu@microsoft.com
David E. Killough
davkill@microsoft.com
MICROSOFT CORPORATION
1 Microsoft Way
Redmond, Washington 98052
Tel. (425) 703-8865

CERTIFICATE OF SERVICE

I HEREBY CERTIFY that on the 25th day of January, 2011, I electronically filed the foregoing document with the Clerk of the Court, using the CM/ECF system, which will automatically send email notification of such filing to all counsel who have entered an appearance in this action.

Respectfully submitted,

/s/ J. Donald Best
J. Donald Best

DATED this 25th day of January 2011.

Exhibit A

US006339780B1

(12) **United States Patent**
Shell et al.

(10) **Patent No.: US 6,339,780 B1**
(45) **Date of Patent: Jan. 15, 2002**

(54) **LOADING STATUS IN A HYPERMEDIA BROWSER HAVING A LIMITED AVAILABLE DISPLAY AREA**

(75) Inventors: **Scott R. Shell**; **Kevin Timothy Shields**, both of Redmond; **Anthony Kitowitz**, Kirkland, all of WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/851,877**

(22) Filed: **May 6, 1997**

(51) **Int. Cl.**^{7} **G06F 17/00**
(52) **U.S. Cl.** **707/526**; 707/102
(58) **Field of Search** 707/1–576; 345/24–440

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,266,253	A	*	5/1981	Matherat	345/24
5,467,459	A	*	11/1995	Alexander et al.	345/514
5,731,813	A	*	3/1998	O'Rourke et al.	345/349
5,760,771	A	*	6/1998	Blonder et al.	345/302
5,774,666	A	*	6/1998	Portuesi	709/218
5,877,766	A	*	3/1999	Bates et al.	345/357
5,973,692	A	*	10/1999	Knowlton et al.	345/348
5,983,005	A	*	11/1999	Monteiro et al.	709/231
6,101,510	A	*	8/2000	Stone et al.	707/513

OTHER PUBLICATIONS

Smallman et al. "Information availability in 2D and 3D displays", IEEE Computer Graphics and Applications, vol. 21 Issue 5, Sep./Oct. 2001, pp. 51–57.*
Hu et al., "Parameterizable fonts based on shape components", IEEE Computer Graphics and Applications, vol. 21 Issue 3, May/Jun. 2001, pp. 70–85.*
Liu et al., "Web–based peer review: the learner as both adapter and reviewer", Education, IEEE Transactions on, vol. 44 Issue 3, Aug. 2001, pp. 246–251.*
www.sciam.com/200/1100issue/1100stjohnbox1.html.*

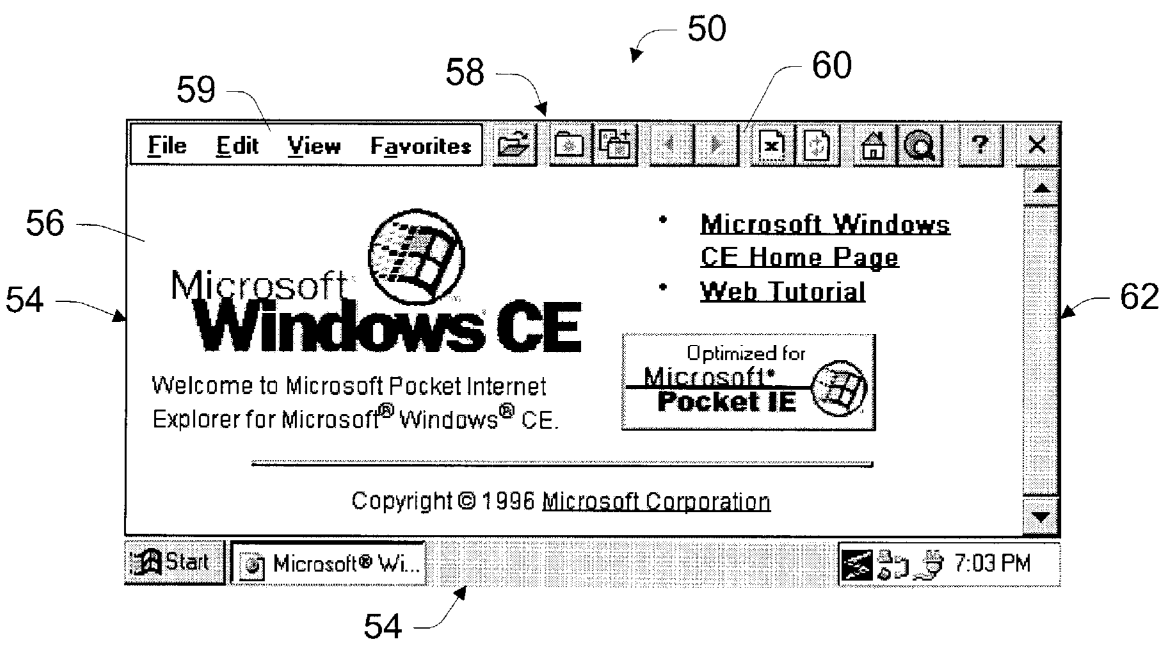
* cited by examiner

Primary Examiner—Thomas Black
Assistant Examiner—David Jung
(74) *Attorney, Agent, or Firm*—Lee & Hayes, PLLC

(57) **ABSTRACT**

Described herein is a portable computer having a limited display area. An Internet or other hypermedia browser executes on the portable computer to load and display content in a content viewing area. During times when the browser is loading content, the browser displays a temporary, animated graphic element over the content viewing area. The graphic element is removed after the content is loaded, allowing unobstructed viewing of the loaded content.

42 Claims, 3 Drawing Sheets



Welcome to Microsoft Pocket Internet Explorer for Microsoft® Windows® CE.

Copyright © 1996 Microsoft Corporation

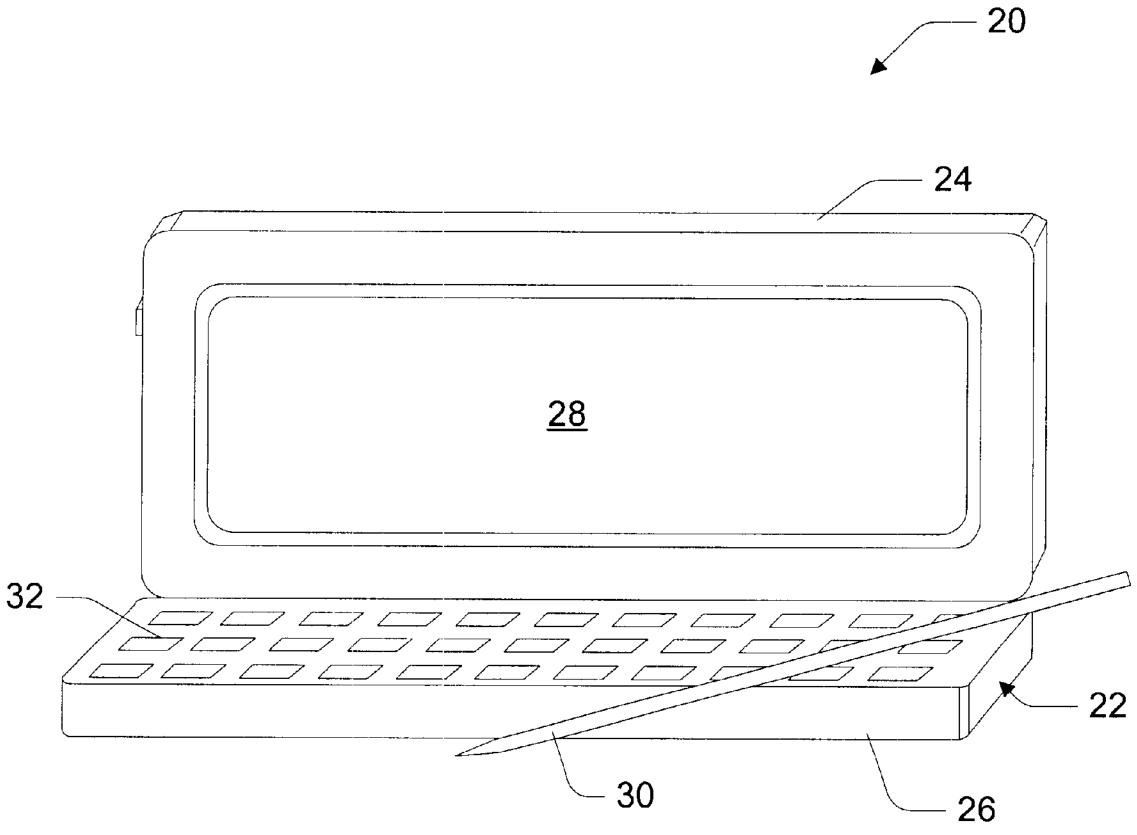


Fig. 1

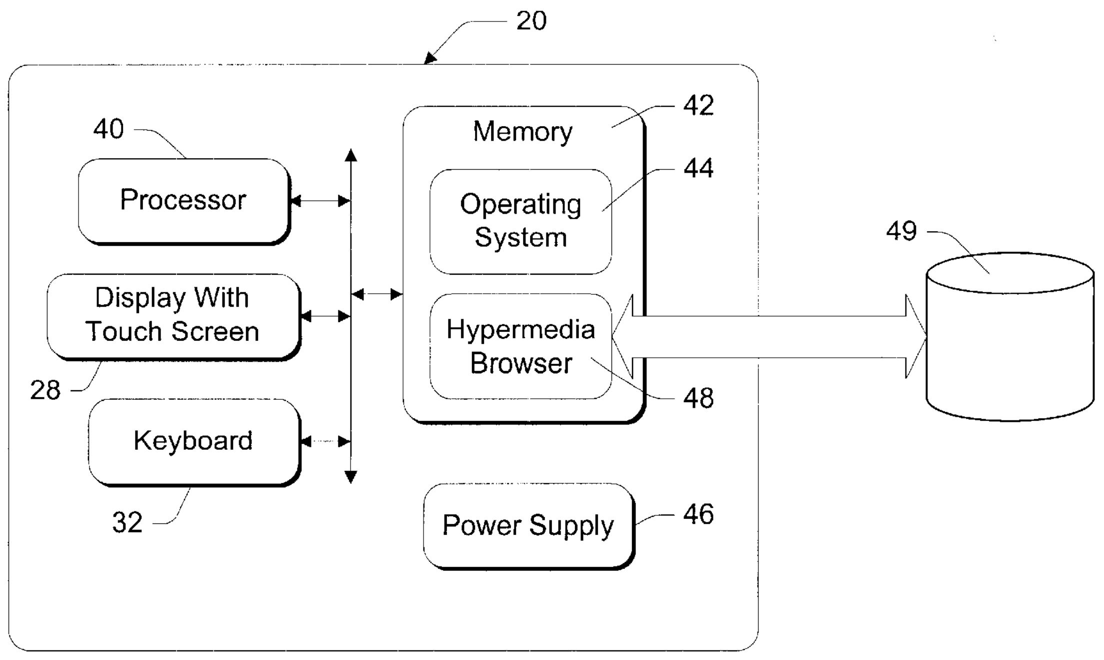
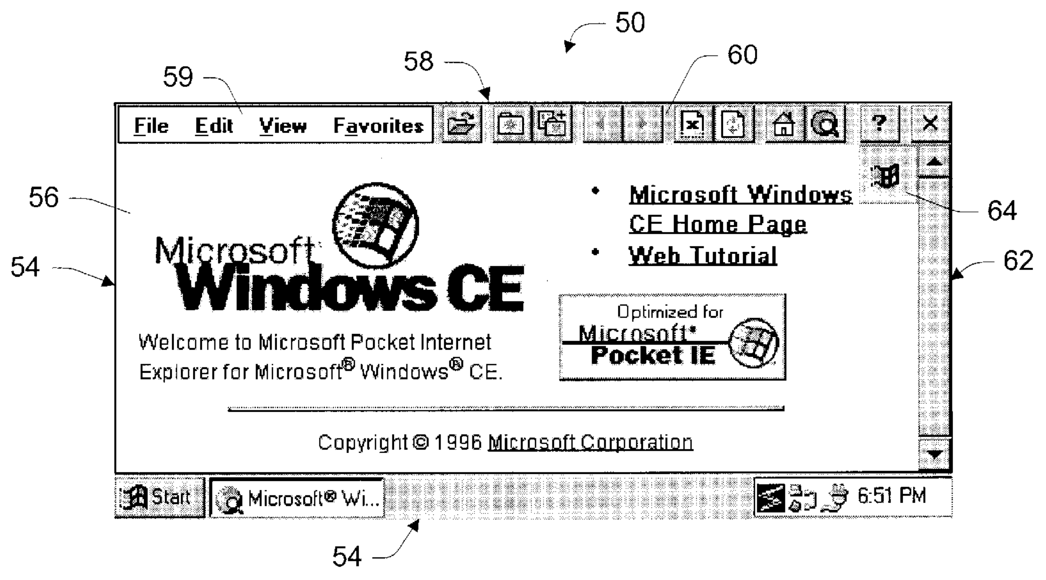


Fig. 2

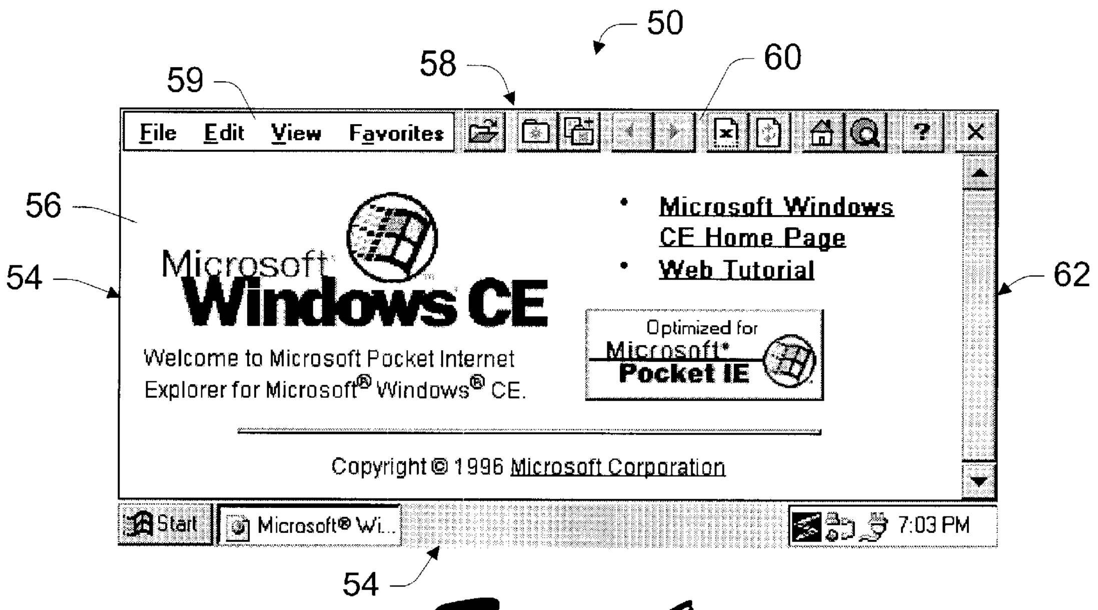


Welcome to Microsoft Pocket Internet Explorer for Microsoft® Windows® CE.

Copyright © 1996 Microsoft Corporation

Fig. 3

Case 3:10-cv-00699-wmc Document #: 37-1 Filed: 01/25/11 Page 5 of 11



Welcome to Microsoft Pocket Internet Explorer for Microsoft® Windows® CE.

Copyright © 1996 Microsoft Corporation

Fig. 4

LOADING STATUS IN A HYPERMEDIA BROWSER HAVING A LIMITED AVAILABLE DISPLAY AREA

TECHNICAL FIELD

This invention relates hypermedia content browsers such as World Wide Web browsers.

BACKGROUND OF THE INVENTION

"Hypermedia" is a metaphor for presenting information in which text, images, sounds, and actions become linked together in a complex, non-sequential web of associations that permit a user to browse through related content and topics, regardless of the presented order of the topics. The term "hypermedia" arises from the similar term "hypertext," which was originally coined to describe the linked text-based documents.

Hypermedia content is widely used for navigation and information dissemination on the "World-Wide Web" (WWW or Web) of the Internet. An application program referred to as a hypermedia browser, hypertext browser, "Web browser" is normally used to retrieve and render hypermedia content from the WWW, although such a browser is also useful for browsing hyperlinked content from other sources.

Hypermedia content is commonly organized as documents with embedded control information. The embedded control information includes formatting specifications, indicating how a document is to be rendered by the Web browser. In addition, such control information can include links or "hyperlinks": symbols or instructions telling the Web browser where to find other related WWW documents. A hyperlink from one hypermedia topic to another is normally established by the author of a hypermedia document, although some applications allow users to insert hyperlinks to desired topics.

A hyperlink is typically rendered by a Web browser as a graphical icon or as highlighted keywords. A user "activates" or "follows" a hyperlink by clicking on or otherwise selecting the icon or highlighted keywords. Activating a link causes the Web browser to load and render the document or resource that is targeted by the hyperlink.

Hyperlink usage is not limited to the Internet. Various multimedia applications and other hypermedia resources utilize hypertext to allow users to navigate through different pieces of information content. For instance, an encyclopedia program might use hyperlinks to provide cross-references to related articles within an electronic encyclopedia. The same program might also use hyperlinks to specify remote information resources such as WWW documents.

Hypermedia browsers have evolved in recent years and are available from several sources. Microsoft's Internet Explorer is one example of a popular browser that is particularly suitable for browsing the WWW and other similar network resources. Browsers such as the Internet Explorer typically have a content viewing area or window, in which textual or other graphical content is displayed. Browser controls such as menus, status displays, and tool icons are located in areas or windows adjacent the viewing area, so that they do not obstruct or interfere with the viewing area.

One persistent characteristic of WWW browsing is that significant delays are often encountered when loading documents and other multimedia content. From the user's perspective, such delays can be quite frustrating. In severe

cases involving long delays, users might be inclined to believe that their browsers have become inoperative. To avoid this situation, browsers typically include some type of status display indicating progress in loading content. In many browsers, this consists of a stationary icon such as a flag or globe that becomes animated during periods when content is being loaded. For instance, such an icon might comprise a flag that is normally stationary but that flutters or waves during content loading. An icon such as this is positioned in a tool area or status area outside of the content viewing area. The icon is visible at all times, but is animated only when content is being loaded.

One very recent development relating to this subject is the emergence of a number of popular, small, handheld computing devices that potentially support Internet browsing. These include palmtops, pocket computers, personal digital assistants, personal organizers, and the like. In this disclosure, this class of computing devices is generally referred to as "handheld personal computers", "handheld PCs", or "H/PCs".

One of the most desirable characteristics of H/PCs is their portability. The compact, portable H/PCs provide a user with real computer-like applications—such as email, PIM (personal information management), spreadsheet, and word processing. Hypermedia browsers are among the application programs available for H/PCs. A traveling user can receive email messages, schedule meetings or appointments, and browse the Internet from the H/PC.

To keep H/PCs small, compromises are of course necessary. Chief among the design compromises is an undersized display. Screen space is very limited. Traditional user interface techniques which users are accustomed to on desktop computers are not available for H/PC displays due to the limited size. Additionally, the screen must be efficiently utilized to enable effective data input from the stylus.

With a hypermedia or Internet browser, in particular, there may not be room enough on the available display to implement an animated status display such as described above.

The inventors, however, have developed a method of implementing a status display even within the limited display areas available on popular H/PCs.

SUMMARY OF THE INVENTION

In accordance with the invention, a browser has a content viewing area that is used for displaying graphical hypermedia content. A temporary, animated graphic element is presented in a corner of the content viewing area during times when the browser is loading content. The graphic element is not displayed during any other times.

BRIEF DESCRIPTION OF THE DRAWINGS

The same reference numbers are used throughout the drawings to reference like components and features.

FIG. **1** is a perspective view of a handheld computing device in an open position.

FIG. **2** is a block diagram of the handheld computing device.

FIGS. **3** and **4** are illustrations of displays generated by a hypermedia browser in accordance with the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. **1** shows a handheld computing device **20**. As used herein, "handheld computing device" means a small computing device having a processing unit that is capable of running one or more application programs, a display, and an input mechanism such as a keypad, a touch-sensitive screen, a track ball, a touch-sensitive pad, a miniaturized QWERTY keyboard, or the like.

The handheld computing device **20** is embodied as a handheld personal computer. The terms "handheld computing device" and "handheld personal computer" (or handheld PC or H/PC) are used interchangeably throughout this disclosure. However, in other implementations, the handheld computing device may be implemented as a personal digital assistant (PDA), a personal organizer, a palmtop computer, a computerized notepad, or the like. The invention can also be implemented in other types of computers and computer-like or computer-controlled devices having graphical display surfaces.

Handheld computing device **20** has a casing **22** with a cover or lid **24** and a base **26**. A liquid crystal display (LCD) **28** with a touch-sensitive screen is mounted to lid **24**. Lid **24** is hinged to base **26** to pivot between an open position, which exposes display **28**, and a closed position, which protects the display. The device is equipped with a stylus **30** to enter data through touchscreen display **28** and a miniature QWERTY keyboard **32**. Stylus **30** and keyboard **32** are both mounted in base **26**. Although the illustrated implementation shows a two-member H/PC **20** with a lid **24** and a base **26**, other implementations of the H/PC might comprise an integrated body without hinged components, as is the case with computerized notepads (e.g., Newton® from Apple Computers).

FIG. **2** shows functional components of the handheld computing device. It has a processor **40**, a computer-readable storage medium or memory **42**, a display **28**, and a keyboard **32**. Memory **42** generally includes both volatile memory (e.g., RAM) and non-volatile memory (e.g., ROM, PCMCIA cards, etc.). The H/PC **20** has a power supply **46** that supplies power to the electronic components. The power supply **46** is preferably implemented as one or more batteries. The power supply **46** might further represent an external power source that overrides or recharges the built-in batteries, such as an AC adapter or a powered docking cradle.

An operating system program **44** is resident in the memory **42** and executes on the processor **40**. The operating system **44** is a multitasking operating system that allows simultaneous execution of multiple applications. The operating system employs a graphical user interface windowing environment that presents applications and documents in specially delineated areas of the display screen called "windows." Each window can act independently, including its own menu, toolbar, pointers, and other controls, as if it were a virtual display device. The handheld computing device may be implemented with other types of operating systems that support a graphical user interface.

The operating system **44** is preferably the Windows® CE operating system from Microsoft Corporation that is configured to execute application programs such as application program **48** shown in FIG. **2**. The Windows® CE operating system is a derivative of Windows® brand operating systems, such as Windows® 95, that is especially designed for handheld computing devices having limited display areas.

In the described embodiment of the invention, application program **48** is an Internet or other hypermedia browser. The browser is stored as a sequence of program instructions in memory **42**, for execution by processor **40**. In other

embodiments, the browser might be stored on a portable or removable type of computer-readable storage medium such as a floppy disk or EPROM (eraseable read-only memory). As used here, the term "hypermedia browser" refers to an application or application program that is capable of displaying or otherwise rendering hypermedia content and of loading additional or alternative hypermedia content in response to a user's selection of hyperlinks.

Browser **48** has access to a hypermedia resource **49**. Often, this resource will be the Internet. However, other sources of hyperlinked content are frequently available and can be efficiently browsed in accordance with the invention. Computer **20** includes a network interface or modem (not shown) for accessing the hypermedia resource.

FIG. **3** shows an example of a graphical display **50** generated by a hypermedia browser **48** in conjunction with operating system **44**. The display includes a number of elements that are generated by making appropriate system calls to the operating system in accordance with well-known protocols. Specifically, Windows® CE supports a subset of the Win32 API set used in the Windows® 95 operating system. These APIs allow an application program to create a variety of on-screen controls with minimal effort.

In this case, the graphical display **50** includes a taskbar **52** presented by the Windows® CE operating system. Browser **48** presents a main window **54**, above taskbar **52**. Browser main window **54** in this example has three primary components. The largest screen area is dedicated to a content viewing area **56**. This is the area in which graphical hypermedia content is displayed.

Content viewing area **56** is bordered along its upper edge by a toolbar **58**. Toolbar **58** is similar in appearance to toolbars used in other application programs designed for the Windows® operating environment, with some characteristics that are unique to the Windows® CE environment. One characteristic that is unique to Windows® CE is that the toolbar includes both a menu area **59** and an icon area **60**. In previous versions of Windows®, these features were presented within their own distinct areas. Another Windows® CE characteristic is that the toolbar is located on what would have been the "title bar" of previous Windows® application programs. The toolbar thus includes an "X" icon **61** that is used to close the browser application. In previous versions of Windows®, the toolbar would have been below or otherwise separate from the title bar.

A scroll bar **62** borders content viewing area **56** along its right side. Scroll bar **62** is used to vertically scroll the content that is presented in content viewing area **56**.

In contrast to prior art hypermedia browsers, browser **48** does not include a permanent "loading status" icon. In fact, no portion of main window **54** is dedicated permanently to displaying loading status. Rather, the browser is configured to display a temporary graphic element **64** over content viewing area **56** during times when the browser is loading content. This temporary graphic element is preferably animated (such as the waving Microsoft® flag shown), and is displayed only when the browser is loading content. It is removed when the browser is not loading content. FIG. **4** shows display **50** after content has been loaded, during a period when no additional content is being loaded. Graphic element **64** has been removed in FIG. **4** because the current Internet page has been completely loaded.

The temporary graphic element is preferably located in a corner of the content viewing area, and obstructs a portion of the viewing area. The upper right corner is preferred because this position is often blank in Internet documents.

The graphic element is created by opening a conventional window in conjunction with the Window® CE windowing operating environment.

This method of displaying loading status achieves the objective of alerting users during periods of time when content is actually being loaded. It does this without requiring a permanent allocation of screen real estate, thus freeing space for other functions. Although there might be some obstruction of hypermedia content, such obstruction is minor and temporary.

The invention has been described primarily in terms of its visual and functional characteristics. However, the invention also includes a method of browsing a hyperlink resource such as the Internet or some other network or data source having linked hypermedia content. The method includes a steps of loading content from the hyperlink resource in response to user selection of hyperlinks contained in said content, and of displaying the content in a content viewing area. The invention also includes a step of displaying a temporary graphic element over the content viewing area during the loading step. The temporary graphic element is removed when content is no longer being loaded.

Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.

What is claimed is:

1. A hypermedia browser embodied on a computer-readable medium for execution on an information processing device having a limited display area, wherein the hypermedia browser has a content viewing area for viewing content and is configured to display a temporary graphic element over the content viewing area during times when the browser is loading content, wherein the temporary graphic element is positioned over the content viewing area to obstruct only part of the content in the content viewing area, wherein the temporary graphic element is not content and wherein content comprises data for presentation which is from a source external to the browser.

2. A hypermedia browser as recited in claim **1**, wherein the browser is configured to display the temporary graphic element over the content viewing area only during times when the browser is loading visible content.

3. A hypermedia browser as recited in claim **1**, wherein the temporary graphic clement indicates to a user that the browser is loading content.

4. A hypermedia browser as recited in claim **1**, wherein the temporary graphic element disappears when the browser's loading of content is complete to indicate to a user that such loading of content is complete.

5. A hypermedia browser as recited in claim **1**, wherein the temporary graphic element is animated.

6. A hypermedia browser as recited in claim **1**, wherein the hypermedia browser displays the temporary graphic element in a corner of the content viewing area.

7. A hypermedia browser as recited in claim **1**, wherein the hypermedia browser presents the temporary graphic element within a temporary window in a windowing operating environment.

8. A hypermedia browser as recited in claim **1**, wherein:

the temporary graphic element is animated; and

the hypermedia browser presents the temporary graphic element within a temporary window in a windowing operating environment.

9. A hypermedia browser as recited in claim **1**, wherein the temporary graphic element conveys status information of the browser.

10. A hypermedia browser of claim **1**, wherein content is data formatted for presentation which is selected from a group consisting of visible effects of a markup language, visible text of such a markup language, and visible results of a scripting language.

11. A hypermedia browser of claim **1**, wherein content is data formatted for presentation which is selected from a group consisting of HTML, text, SGML, XML, java, XHTML, JavaScript, streaming video, VRML, Active X, Flash. scripting language for the world wide web.

12. An information processing device comprising:

a processor;

a display;

a hypermedia browser executing on the processor to load and display content in a content viewing area on the display;

wherein the hypermedia browser displays a temporary graphic element over the content viewing area during times when the browser is loading visible content;

wherein the temporary graphic element is positioned only over a portion of the content viewing area and obstructs only part of the visible content in the content viewing area; and

wherein the temporary graphic element indicates to a user that the browser is loading content and content comprises data for presentation which is from a source external to the browser.

13. An information processing device as recited in claim **12**, wherein the temporary graphic element is animated.

14. An information processing device as recited in claim **12**, wherein the hypermedia browser displays the temporary graphic element in a corner of the content viewing area.

15. An information processing device as recited in claim **12**, wherein the hypermedia browser displays the temporary graphic element within a temporary window in a windowing operating environment.

16. An information processing device as recited in claim **12**, wherein:

the temporary graphic element is animated; and

the hypermedia browser displays the temporary graphic element within a temporary window in a windowing operating environment.

17. A hypermedia browser of claim **12**, wherein content is data formatted for presentation which is selected from a group consisting of visible effects of a markup language, visible text of such a markup language, and visible results of a scripting language.

18. A hypermedia browser of claim **12**, wherein content is data formatted for presentation which is selected from a group consisting of HTML, text, SGML, XML, java, XHTML, JavaScript, streaming video, VRML, Active X, Flash. scripting language for the world wide web.

19. A method of browsing a hyperlink resource, comprising the following steps:

loading content from the hyperlink resource in response to user selection of hyperlinks contained in said content;

displaying the content in a content viewing area;

displaying a temporary graphic element over the content viewing area during the loading step, wherein the temporary graphic element obstructs only part of the content in the content viewing area;

wherein the loading, the content displaying, and the temporary graphic element displaying steps occur at least partially concurrently; and wherein content comprises data for presentation which is from a source external to the browser.

20. An information processing device as recited in claim **12**, wherein the temporary graphic element is not content.

21. An information processing device as recited in claim **12**, wherein the temporary graphic element disappears when the browser's loading of content is complete to indicate to a user that such loading of content is complete.

22. A method as recited in claim **19**, wherein the temporary graphic element is not content.

23. A method as recited in claim **19**, wherein the temporary graphic element indicates to a user that the loading step is being performed.

24. A method as recited in claim **19**, further comprising removing the temporary graphic element once the loading step is complete to indicate to a user that the loading step is complete.

25. A method as recited in claim **19**, further comprising an additional step of animating the temporary graphic element.

26. A method as recited in claim **19**, wherein the displaying step includes displaying the temporary graphic element in a corner of the content viewing area.

27. A method as recited in claim **19**, wherein the displaying step includes displaying the temporary graphic element within a temporary window in a windowing operating environment.

28. A method as recited in claim **19**, further comprising an additional step of animating the temporary graphic element, wherein the displaying step includes displaying the temporary graphic element within a temporary window in a windowing operating environment.

29. A computer-readable storage medium containing instructions that are executable for performing the steps recited in claim **19**.

30. A hypermedia browser of claim **19**, wherein content is data formatted for presentation which is selected from a group consisting of visible effects of a markup language, visible text of such a markup language, and visible results of a scripting language.

31. A hypermedia browser of claim **19**, wherein content is data formatted for presentation which is selected from a group consisting of HTML, text, SGML, XML, java, XHTML, JavaScript, streaming video, VRML, Active X, Flash. scripting language for the world wide web.

32. A method of indicating a content "load status" of a hypermedia browser having a content viewing area for viewing content, the method comprising:

displaying loaded content within the content viewing area of a screen of a hypermedia browser, the screen being without a "load status" graphic element, wherein a "load status" graphic element indicates a current content load status of the hypermedia browser;

receiving an instruction to load new content into the content viewing area;

loading such new content into the content viewing area; and

while loading, displaying a "load status" graphic element over the content viewing area so that the graphic element obstructs only part of the content in such content viewing area; and

wherein content comprises data for presentation which is from a source external to the browser.

33. A method as recited in claim **32** further comprising, upon completion of the loading, removing the "load status" graphic element to reveal the part of the content in the

content viewing area that the graphic element obstructed when the element was displayed.

34. A hypermedia browser of claim **32**, wherein content is data formatted for presentation which is selected from a group consisting of visible effects of a markup language, visible text of such a markup language, and visible results of a scripting language.

35. A hypermedia browser of claim **32**, wherein content is data formatted for presentation which is selected from a group consisting of HTML, text, SGML, XML, java, XHTML, JavaScript, streaming video, VRML, Active X, Flash. scripting language for the world wide web.

36. A computer-readable medium having computer-executable instructions that, when executed by a computer, perform a method of indicating a content "load status" of a hypermedia browser having a content viewing area for viewing content, the method comprising:

displaying loaded content within the content viewing area of a screen of a hypermedia browser, the screen is without a "load status" graphic element, wherein a "load status" graphic element indicates a current content load status of the hypermedia browser;

receiving an instruction to load new content into the content viewing area;

loading such new content into the content viewing area; and

while loading, displaying a "load status" graphic element over the content viewing area so that the graphic element obstructs only part of the content in such content viewing area; and

wherein content comprises data for presentation which is from a source external to the browser.

37. A hypermedia browser of claim **36**, wherein content is data formatted for presentation which is selected from a group consisting of visible effects of a markup language, visible text of such a markup language, and visible results of a scripting language.

38. A hypermedia browser of claim **36**, wherein content is data formatted for presentation which is selected from a group consisting of HTML, text, SGML, XML, java, XHTML, JavaScript, streaming video, VRML, Active X, Flash. scripting language for the world wide web.

39. A computer-readable medium as recited in claim **36** further having additional computer-executable instructions that perform a method comprising, upon completion of the loading, removing the "load status" graphic element to reveal the part of the content in the content viewing area that the graphic element obstructed when the element was displayed.

40. An information processing device comprising:

a processor;

a display;

a hypermedia browser executing on the processor to load and display content in a content viewing area on the display;

wherein the hypermedia browser is configured to operate in a content-loading mode and a content-loaded mode;

in the content-loaded mode, the hypermedia browser displays loaded content in the content viewing area and no "load status" graphic element is displayed, wherein absence of such "load status" graphic element indicates that the browser is in the content-loaded mode;

in the content-loading mode, the hypermedia browser loads content, displays such content in the content viewing area as it loads, and displays a "load status" graphic element over the content view area obstructing part of the content displayed in the content viewing area, wherein presence of such "load status" graphic element indicates that the browser is in the content-loading mode; and

wherein content comprises data for presentation which is from a source external to the browser.

41. A hypermedia browser of claim **40**, wherein content is data formatted for presentation which is selected from a group consisting of visible effects of a markup language, visible text of such a markup language, and visible results of a scripting language.

42. A hypermedia browser of claim **40**, wherein content is data formatted for presentation which is selected from a group consisting of HTML, text, SGML, XML, java, XHTML, JavaScript, streaming video, VRML, Active X, Flash. scripting language for the world wide web.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO.	: 6,339,780 B1	Page 1 of 1
DATED	: January 15, 2002	
INVENTOR(S)	: Shell et al.	

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 5,
Line 15, change "steps" to -- step --.

Signed and Sealed this

Seventeenth Day of September, 2002

Attest:

[illegible]

JAMES E. ROGAN

Attesting Officer

Director of the United States Patent and Trademark Office

Case: 3:10-cv-00699-wmc Document #: 37-2 Filed: 01/25/11 Page 1 of 20

Exhibit B

US007411582B2

(12) **United States Patent**
Toepke et al.

(10) **Patent No.: US 7,411,582 B2**
(45) **Date of Patent: *Aug. 12, 2008**

(54) SOFT INPUT PANEL SYSTEM AND METHOD

(75) Inventors: **Michael G. Toepke**, Bellevue, WA (US); **Jeffrey R. Blum**, Seattle, WA (US); **Kathryn L. Parker**, Fall City, WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **10/989,877**

(22) Filed: **Nov. 15, 2004**

(65) **Prior Publication Data**

US 2005/0088421 A1 Apr. 28, 2005

Related U.S. Application Data

(63) Continuation of application No. 10/072,111, filed on Feb. 8, 2002, now Pat. No. 6,819,315.

(51) **Int. Cl.**
G06F 3/041 (2006.01)

(52) **U.S. Cl.** **345/173**; 345/179; 345/905; 715/825

(58) **Field of Classification Search** 345/173, 345/179, 347, 762, 156, 901, 905, 87; 715/808, 715/825–829; 395/340
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,058,046	A	10/1991	Lapeyre
5,128,672	A	7/1992	Kaehler
5,252,951	A	10/1993	Tannenbaum et al.

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0464712	1/1992

(Continued)

OTHER PUBLICATIONS

"Function-independent Approach to Driving Soft Keyboards," IBM Technical Disclosure Bulletin, vol. 33, No. 4, pp. 159-161 (Sep. 1, 1990).

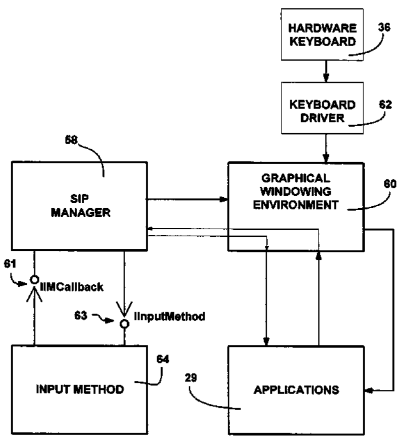
(Continued)

Primary Examiner—Richard Hjerpe
Assistant Examiner—Kimnhung Nguyen

(57) **ABSTRACT**

A method and system for receiving user input data into a computer system having a graphical windowing environment. A touch-sensitive display screen for displaying images and detecting user activity is provided. A management component connects to the graphical windowing environment to create an input panel window for display on the screen. An input method which may be a COM object is selected from multiple input methods available, and installed such that the input method can call functions of the management component. Each input method includes a corresponding input panel, such as a keyboard, which it draws in the input panel window. When the user taps the screen at the input panel, the input method calls a function of the management component to pass corresponding input information appropriate information such as a keystroke or character to the management component. In response, the management component communicates the user data to the graphical windowing environment as a message, whereby an application program receives the message as if the message was generated on a hardware input device.

31 Claims, 8 Drawing Sheets



U.S. PATENT DOCUMENTS

RE34,476	E		12/1993	Norwood	
5,276,794	A		1/1994	Lamb, Jr.	
5,517,578	A		5/1996	Altman et al.	
5,528,743	A		6/1996	Tou et al.	
5,574,482	A		11/1996	Niemeier	
5,596,702	A		1/1997	Stucka et al.	395/40
5,644,339	A		7/1997	Mori et al.	345/173
5,748,512	A		5/1998	Vargas	
5,760,773	A	*	6/1998	Berman et al.	715/808
5,777,605	A		7/1998	Yoshinobu et al.	
5,781,181	A		7/1998	Yanai et al.	
5,818,425	A		10/1998	Want et al.	
5,838,302	A		11/1998	Kuriyama et al.	
5,914,707	A	*	6/1999	Kono	345/173
5,936,614	A		8/1999	An et al.	
5,956,423	A		9/1999	Frink et al.	382/187
6,008,799	A		12/1999	Van Kleeck	
6,018,335	A		1/2000	Onley	
6,031,525	A		2/2000	Perlin	
6,069,628	A		5/2000	Farry et al.	345/348
6,094,197	A		7/2000	Buxton et al.	
6,295,052	B1		9/2001	Kato et al.	
6,819,315	B2	*	11/2004	Toepke et al.	345/173

FOREIGN PATENT DOCUMENTS

JP	01-191226	8/1989
JP	06-324806	11/1994
JP	08-22385	1/1996
JP	08-328805	12/1996
WO	WO 9209944	6/1992

OTHER PUBLICATIONS

"Soft Adaptive Follow-Finger Keyboard for Touch-Screen Pads," IBM Technical Disclosure Bulletin, vol. 36, No. 11, pp. 5-7, (Nov. 1, 1993).

Kano, Nadine, *Developing International Software for Windows 95 and Windows NT,* Chapter 7, Appendix N and Appendix O, Microsoft Press, pp. 202-229, 553-556, 557-563.

International Search Report in Corresponding PCT Application No. PCT/US98/26683.

* cited by examiner

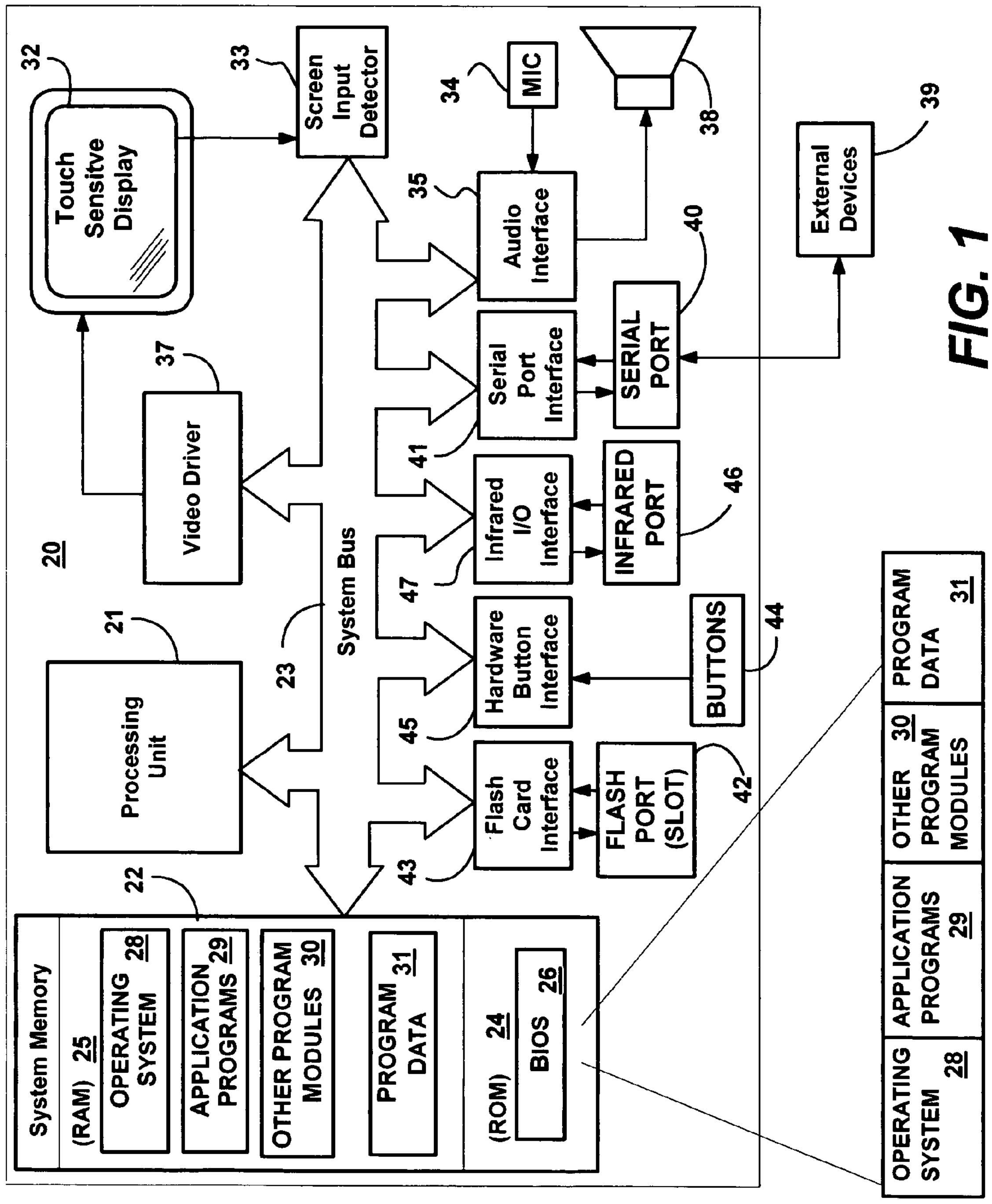

20
Touch Sensitve Display
32
Screen Input Detector
33
MIC
34
38
Audio Interface
35
Video Driver
37
Serial Port Interface
SERIAL PORT
40
External Devices
39
41
Infrared I/O Interface
INFRARED PORT
46
47
Processing Unit
21
System Bus
23
Hardware Button Interface
BUTTONS
44
45
Flash Card Interface
FLASH PORT (SLOT)
42
43
22
System Memory
(RAM) 25
OPERATING SYSTEM 28
APPLICATION PROGRAMS 29
OTHER PROGRAM MODULES 30
PROGRAM DATA 31
(ROM) 24
BIOS 26
OPERATING SYSTEM 28
APPLICATION PROGRAMS 29
OTHER PROGRAM MODULES 30
PROGRAM DATA 31

FIG. 1

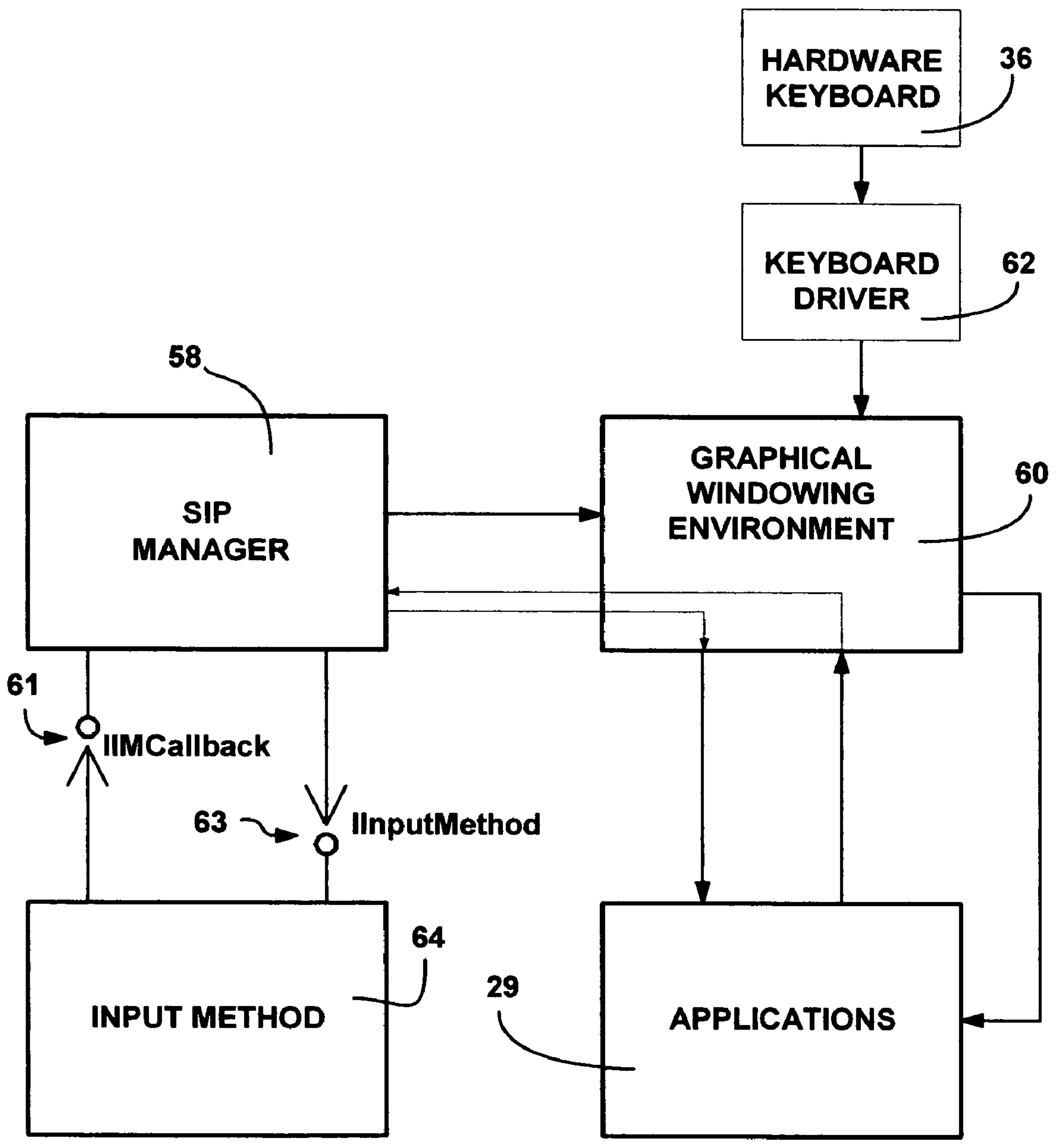

HARDWARE KEYBOARD
36
KEYBOARD DRIVER
62
58
SIP MANAGER
GRAPHICAL WINDOWING ENVIRONMENT
60
61
IIMCallback
63
IInputMethod
64
INPUT METHOD
29
APPLICATIONS

FIG. 2

Case: 3:10-cv-00699-wmc Document #: 37-2 Filed: 01/25/11 Page 6 of 20

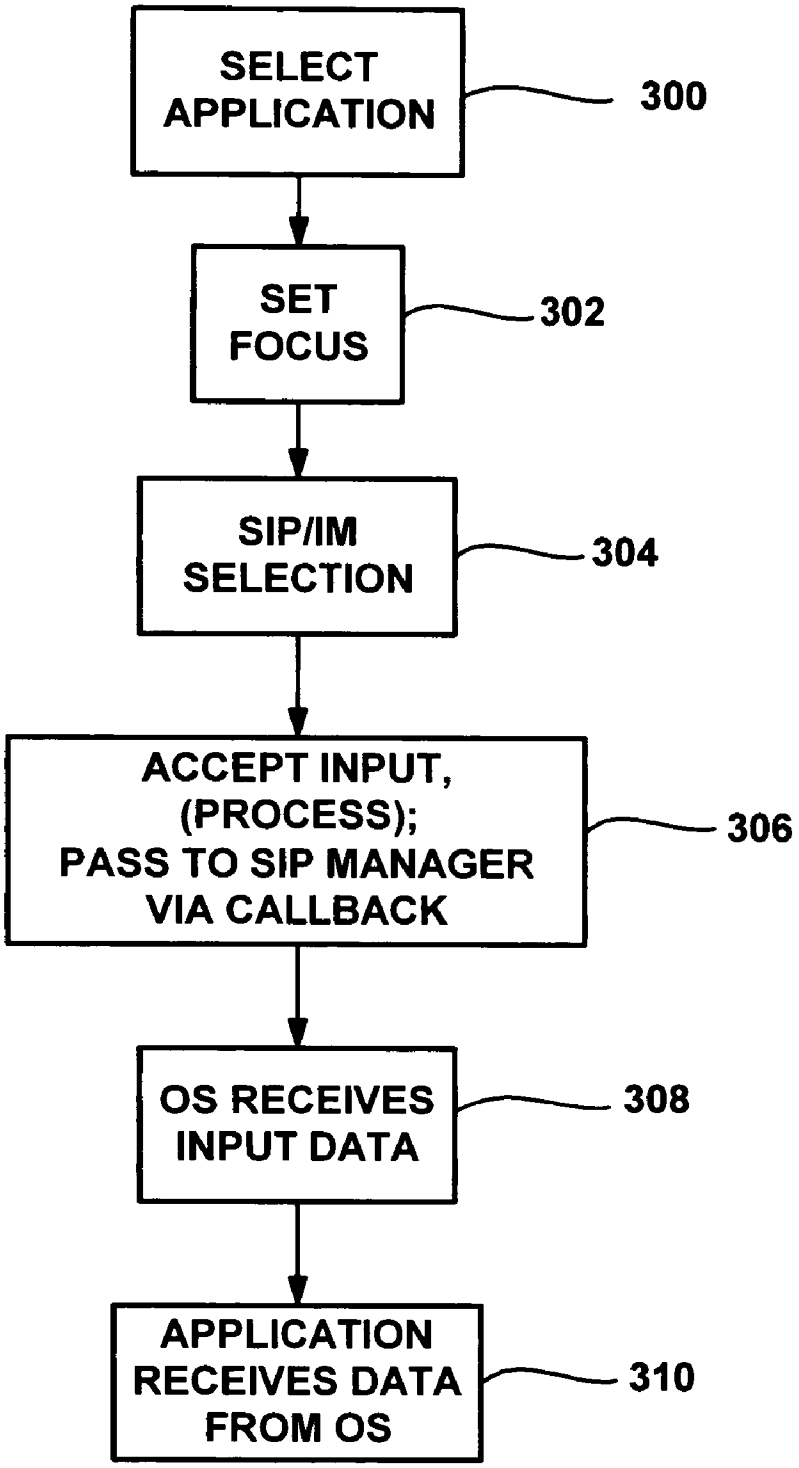

SELECT APPLICATION
300
SET FOCUS
302
SIP/IM SELECTION
304
ACCEPT INPUT, (PROCESS); PASS TO SIP MANAGER VIA CALLBACK
306
OS RECEIVES INPUT DATA
308
APPLICATION RECEIVES DATA FROM OS
310

FIG. 3

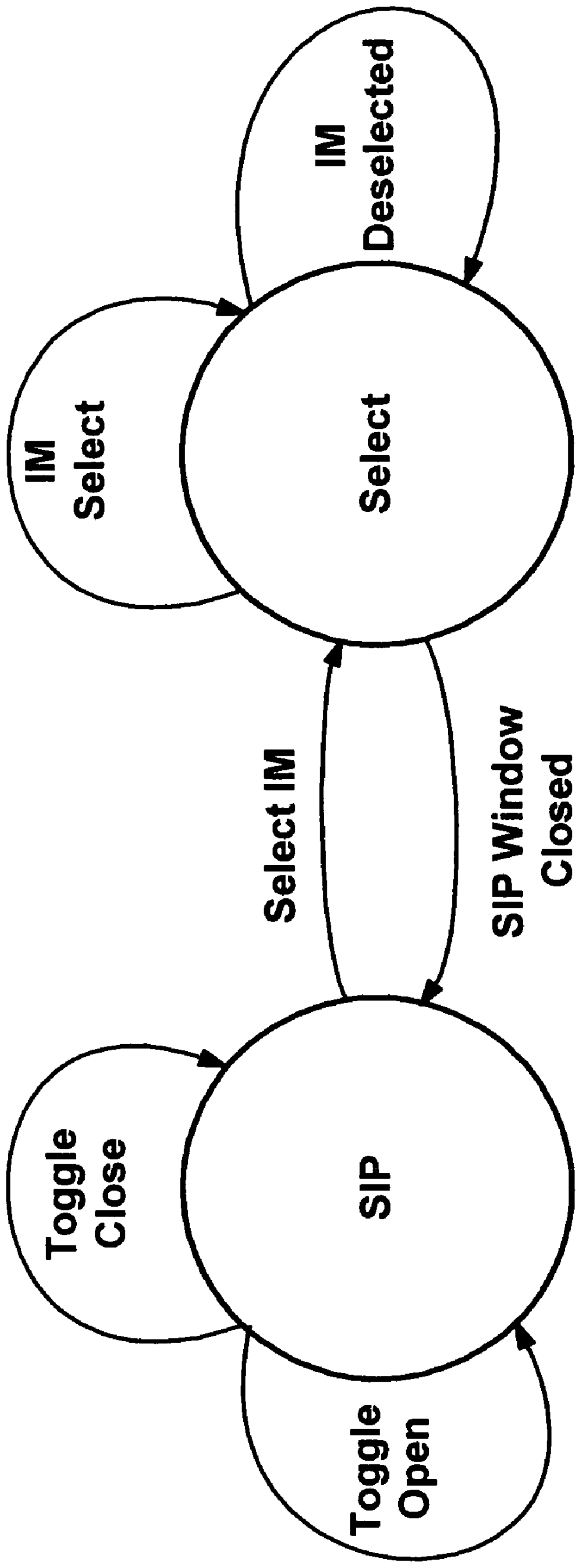

IM Deselected
IM Select
Select
Select IM
SIP Window Closed
Toggle Close
SIP
Toggle Open

FIG. 4

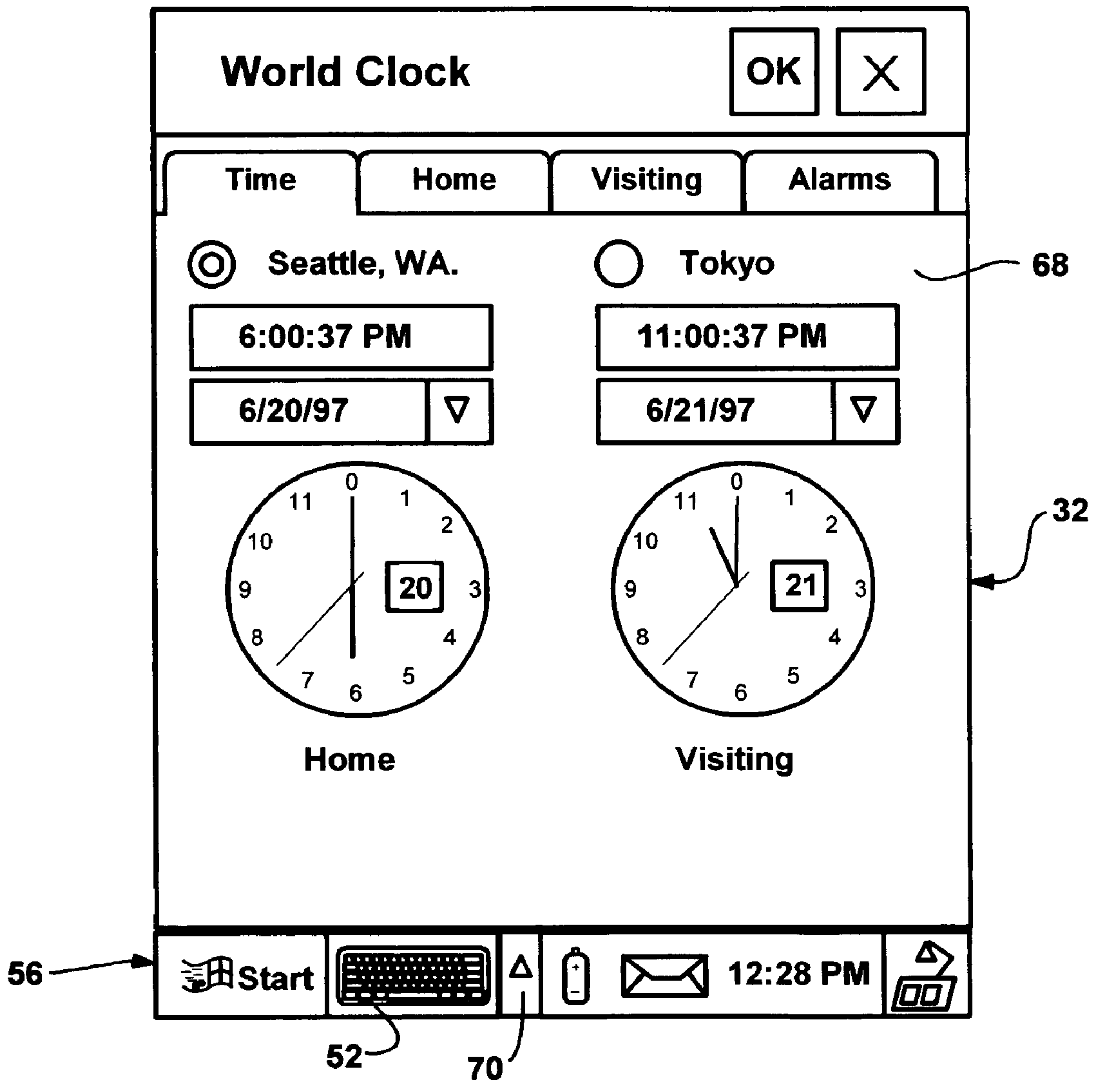

World Clock
OK
X
Time
Home
Visiting
Alarms
Seattle, WA.
Tokyo
68
6:00:37 PM
11:00:37 PM
6/20/97
6/21/97
32
20
21
Home
Visiting
56
Start
12:28 PM
52
70

FIG. 5

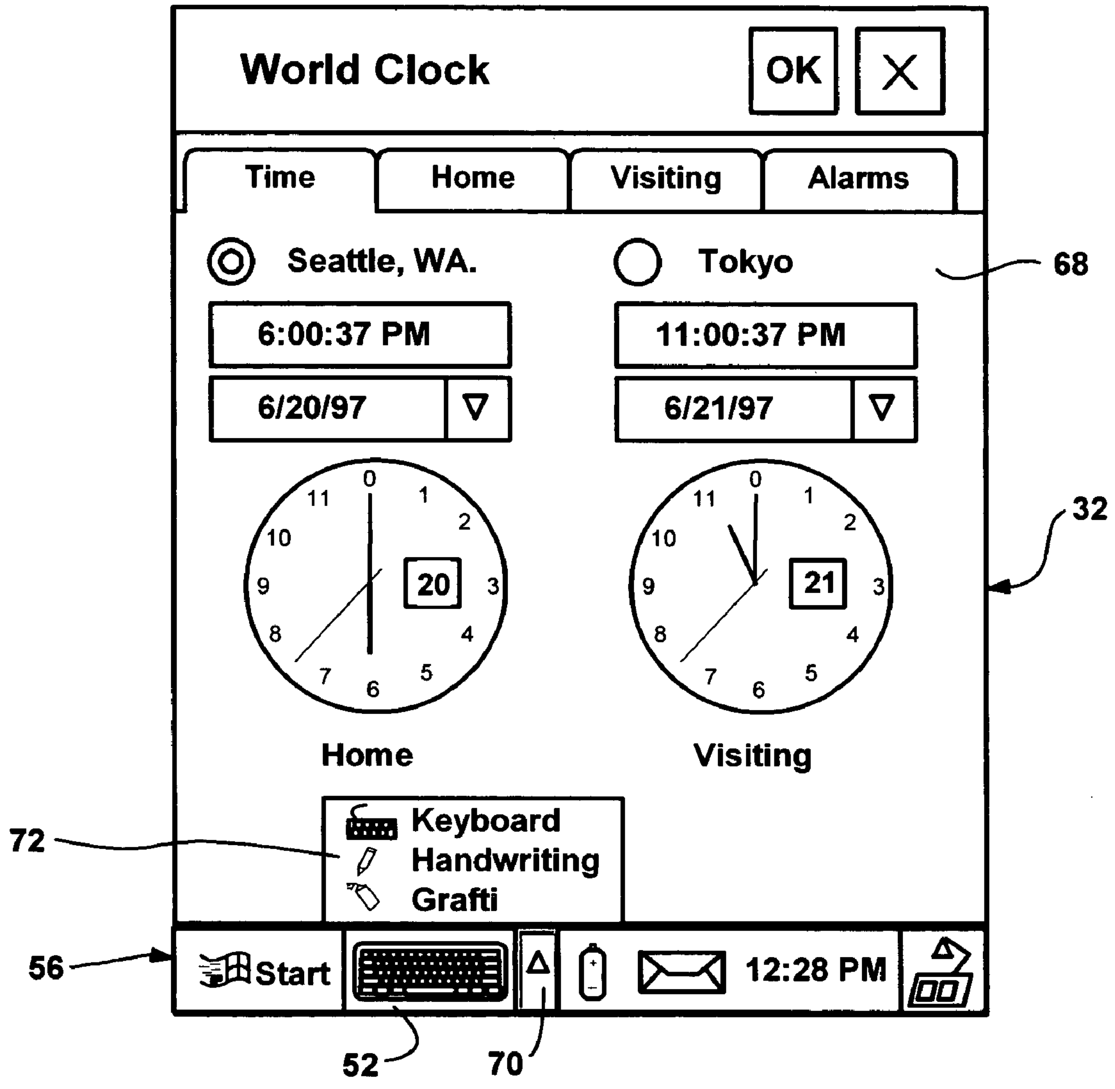

World Clock
OK
X
Time
Home
Visiting
Alarms
Seattle, WA.
Tokyo
68
6:00:37 PM
11:00:37 PM
6/20/97
6/21/97
0
1
2
3
4
5
6
7
8
9
10
11
20
21
32
Home
Visiting
Keyboard
Handwriting
Grafti
72
56
Start
12:28 PM
52
70

FIG. 6

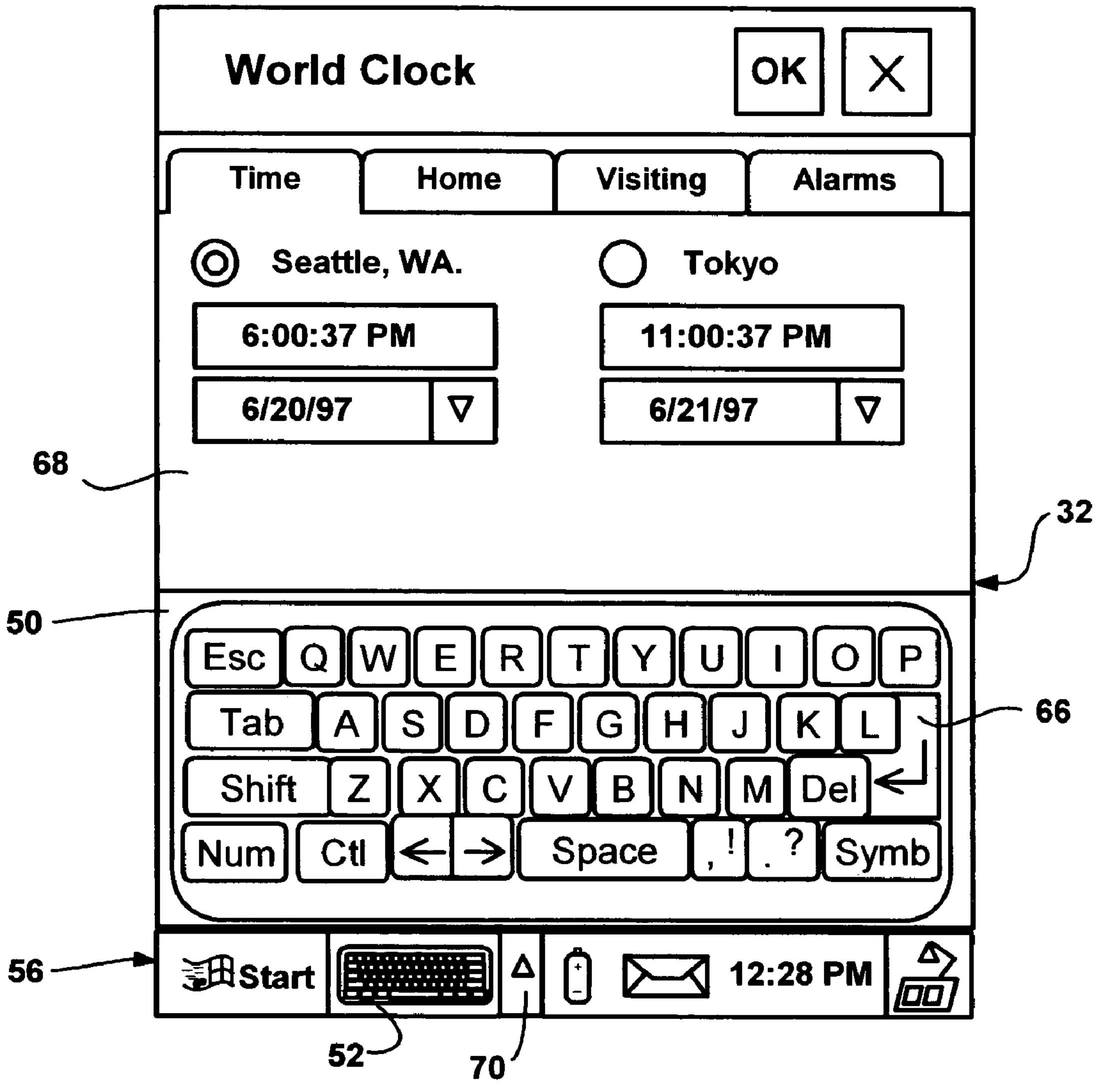

World Clock
OK
X
Time
Home
Visiting
Alarms
Seattle, WA.
Tokyo
6:00:37 PM
11:00:37 PM
6/20/97
6/21/97
Esc Q W E R T Y U I O P
Tab A S D F G H J K L
Shift Z X C V B N M Del
Num Ctl Space , ! . ? Symb
Start
12:28 PM
68
32
50
66
56
52
70

FIG. 7

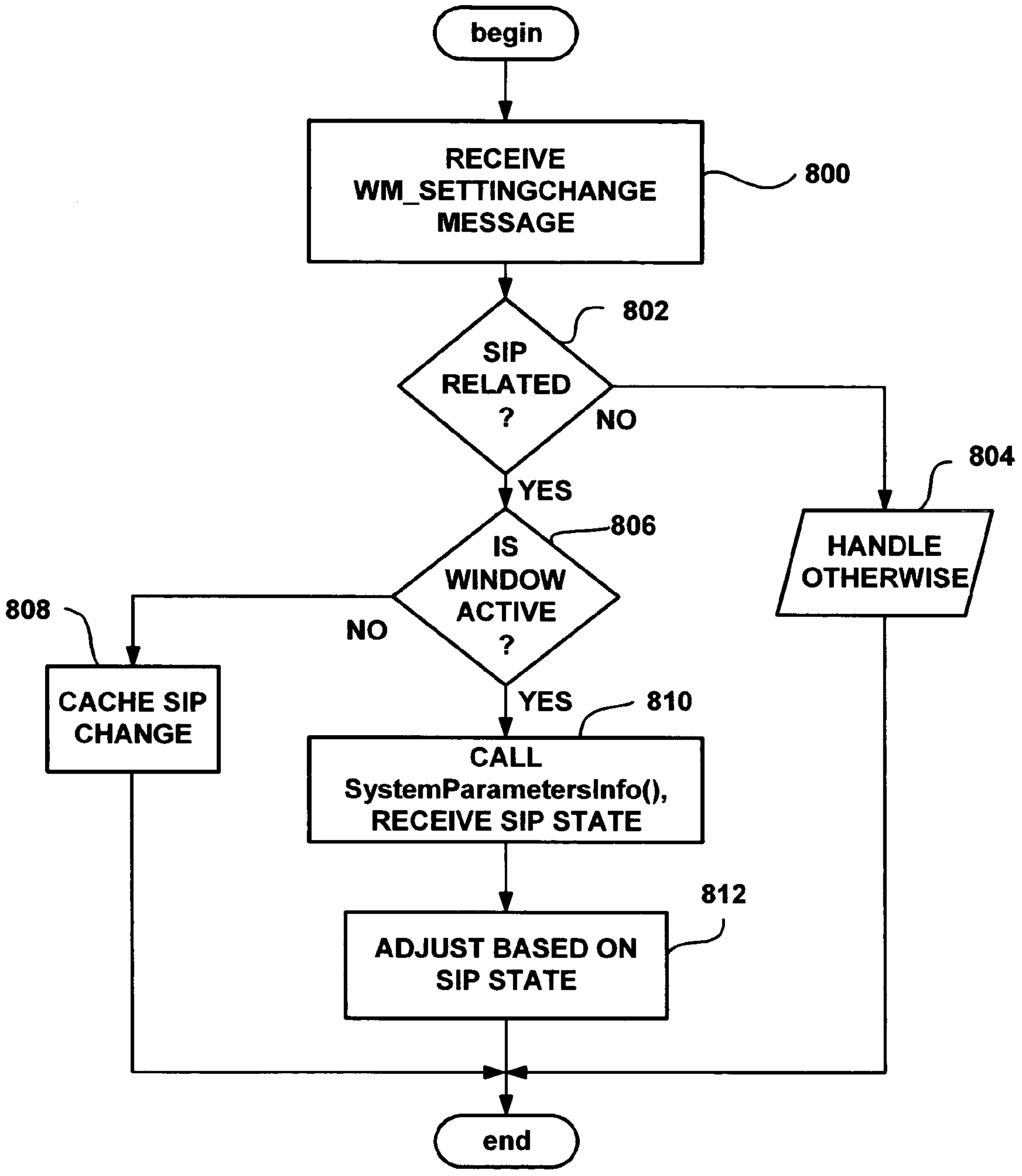

begin
RECEIVE WM_SETTINGCHANGE MESSAGE
800
802
SIP RELATED ?
NO
YES
804
HANDLE OTHERWISE
806
IS WINDOW ACTIVE ?
808
NO
CACHE SIP CHANGE
YES
810
CALL SystemParametersInfo(), RECEIVE SIP STATE
812
ADJUST BASED ON SIP STATE
end

FIG. 8

SOFT INPUT PANEL SYSTEM AND METHOD

CROSS-REFERENCE TOo RELATED APPLICATION

This is a continuation of U.S. patent application Ser. No. 10/072,111 filed Feb. 8, 2002, which is a continuation of U.S. patent application Ser. No. 08/991,277 filed Dec. 16, 1997.

FIELD OF THE INVENTION

The invention relates generally to computer systems, and more particularly to the input of data into a computer system.

BACKGROUND OF THE INVENTION

Small, mobile computing devices such as personal desktop assistants including hand-held and palm-top computers and the like are becoming important and popular user tools. In general, they are becoming small enough to be extremely convenient while consuming less and less battery power, and at the same time becoming capable of running more and more powerful applications.

Although such devices continue to shrink in size, size limitations are being reached as a result of human limitations. For example, a full character keyboard that enables user data input cannot be so small that human fingers cannot depress the individual keys thereon. As a result, the size of such devices (e.g., palm-top computers) has become limited to that which can accommodate a full character keyboard for an average user.

One solution to reducing the size of the portion of the device that receives user input is to provide a touch-sensitive display, and thereby substantially eliminate the need for a physical keyboard. To this end, an application program such as a word processor displays a keyboard, whereby the user enters characters by touching the screen at locations corresponding to the displayed keys. Of course, touch screen devices can also be used simultaneously with devices having a physical keyboard, whereby characters can also be entered by manually pressing the keys of the physical keyboard.

While a touch-screen device serves to provide a suitable means of user data entry, the data entry panel is typically part of the application program, i.e., each application needs to develop its own touch-sensitive interface. As a result, a substantial amount of duplication takes place. For example, both the word processor and a spreadsheet program require alphanumeric keyboard input, whereby each provides its own touch-screen keyboard interface. Other types of programs, such as a calculator program, need a numeric keypad with additional keys representing mathematical operations. This makes each program larger, more complex and consumes computer system resources.

Alternatively, the operating system can supply all the virtual keyboards and thus eliminate the redundancy, however this limits applications to using only those virtual keyboards supplied by the operating system. Newer applications (e.g., those added by plug-in modules) are unable to provide an input mechanism that is more tailored to its particular needs. For example, a new paintbrush program may need its own graphical input screen. In sum, there is a tradeoff between flexibility and efficiency that is inherent with present user data input mechanisms.

OBJECTS AND SUMMARY OF THE INVENTION

Accordingly, it is an object of the present invention to provide an improved method system for entering user data into a computer system.

Another object of the present invention is to provide the method and system for user data entry that is both efficient and flexible.

In accomplishing those objects, it is a related object to provide a method and system of the above kind that functions with touch-sensitive input mechanisms.

Yet another object is to provide a method and system as characterized above that enables a plurality of applications to receive user input from a common input method.

A related object is to provide a method and system that enables selection of one or more input methods for each application from among a set of interchangeable input methods.

Yet another object is to provide such a method and system that is cost-effective, reliable, extensible and simple to implement.

Briefly, the present invention provides a method and system for receiving user data input into a computer system, such as a computer system having a graphical windowing environment. The invention may utilize a touch-sensitive display screen for displaying images and detecting user contact therewith (or proximity thereto). A management component operatively connected to the graphical windowing environment creates an input panel window for display on the screen. An input method is selected from among a plurality of such input methods and installed, whereby the input method can call functions of the management component. Each input method includes a corresponding input panel, such as a keyboard, which it draws in the input panel window. When user data is received via the input panel, the input method calls a function of the management component to pass the user data thereto, and in response, the management component communicates the user data to the graphical windowing environment such as in a windows message. An application program receives the message, such as corresponding to a keystroke, as if the message was generated on a hardware keyboard.

Other objects and advantages will become apparent from the following detailed description when taken in conjunction with the drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram representing a computer system into which the present invention may be incorporated;

FIG. **2** is a block diagram representing various components and connections therebetween for implementing interchangeable input panels according to an aspect of the present invention;

FIG. **3** is a flow diagram generally representing a process for getting user input from a selected input method to a selected application in accordance with one aspect of the present invention;

FIG. **4** is a state diagram generally representing SIP selection states;

FIG. **5** represents a display on a touch-sensitive display screen on an exemplary computing device;

FIG. **6** represents a display on a touch-sensitive display screen on an exemplary computing device providing the ability to select from among interchangeable input panels in accordance with the present invention;

FIG. **7** represents a display on a touch-sensitive display screen wherein a keyboard has been selected as an input panel in accordance with the present invention; and

FIG. **8** is a flow diagram representing the general steps taken in response to a change in SIP status.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Exemplary Operating Environment

FIG. **1** and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by a hand-held computing device such as a personal desktop assistant. Generally, program modules include routines, programs, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types.

Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including palm-top, desktop or laptop personal computers, mobile devices such as pagers and telephones, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to FIG. **1**, an exemplary system for implementing the invention includes a general purpose computing device in the form of a hand-held personal computing device **20** or the like, including a processing unit **21**, a system memory **22**, and a system bus **23** that couples various system components including the system memory to the processing unit **21**. The system bus **23** may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes read-only memory (ROM) **24** and random access memory (RAM) **25**. A basic input/output system **26** (BIOS), containing the basic routines that help to transfer information between elements within the hand-held computer **20**, such as during start-up, is stored in the ROM **24**.

A number of program modules are stored in the ROM **24** and/or RAM **25**, including an operating system **28** (preferably Windows CE), one or more application programs **29**, other program modules **30** and program data **31**. A user may enter commands and information into the hand-held computer **20** through input devices such as a touch-sensitive display screen **32** with suitable input detection circuitry **33**. Other input devices may include a microphone **34** connected through a suitable audio interface **35** and a physical (hardware) keyboard **36** (FIG. **2**). The output circuitry of the touch-sensitive display **32** is also connected to the system bus **23** via video driving circuitry **37**. In addition to the display **32**, the device may include other peripheral output devices, such as at least one speaker **38** and printers (not shown).

Other external input or output devices **39** such as a joystick, game pad, satellite dish, scanner or the like may be connected to the processing unit **21** through an RS-232 or the like serial port **40** and serial port interface **41** that is coupled to the system bus **23**, but may be connected by other interfaces, such as a parallel port, game port or universal serial bus (USB). The hand-held device **20** may further include or be capable of connecting to a flash card memory (not shown) through an appropriate connection port (e.g., slot) **42** and interface **43**. A number of hardware buttons **44** such as switches, buttons (e.g., for switching application) and the like may be further provided to facilitate user operation of the device **20**, and are also connected to the system via a suitable interface **45**. An infrared port **46** and corresponding interface/driver **47** are provided to facilitate communication with other peripheral devices, including other computers, printers, and so on (not shown). It will be appreciated that the various components and connections shown are exemplary and other components and means of establishing communications links may be used.

Soft Input Panel

The soft input panel architecture is primarily designed to enable character, key-based and other user data input via the touch screen **32** of the device **20** rather than a physical keyboard **36**. However, as can be appreciated, a given computer system **20** may optionally and additionally include a physical keyboard, as represented by the dashed box **36** of FIG. **2**. Moreover, as will become apparent, the "soft input panel" need not be an actual touch-sensitive panel arranged for directly receiving input, but may alternatively operate via another input device such as the microphone **34**. For example, spoken words may be received at the microphone **34**, recognized, and displayed as text in an on-screen window, i.e., a soft input panel.

FIG. **2** shows a block diagram implementing the SIP architecture in accordance with one aspect of the present invention. The computer system **20** includes an operating system (OS) **28** such as the graphical windowing environment **60**. Such a graphical windowing environment **60** is generally operational to receive user input through a variety of devices including the keyboard **36**, a mouse (not shown), a digitizer (not shown) and so on. In turn, the graphical windowing environment **60** may provide such user input to an application having "input focus," typically in the form of a keyboard character event. Note that a number of applications **29** may be executable by the computer system, however one application that is currently running is said to have "input focus" and receive the input.

In accordance with one aspect of the present invention, the present architecture employs a SIP manager **58** to provide a single and flexible interface for a plurality of different input methods **64**. In general, the SIP manager **58** provides keystrokes from a selected input method **64** to the graphical windowing environment **60** (e.g., the Windows CE operating system **28**). Once received, the graphical windowing environment **60** sends information corresponding to the user input data to an application **29** (i.e., the application whose window currently has input focus) in the form of that keystroke, mouse or other message placed in the message queue of the application's window. The passing of such messages is well known in Windows programming and is described in "*Programming Windows* 95," Charles Petzold, Microsoft Press (1996), hereby incorporated by reference. As a result, any application capable of handling keyboard input may be used with any appropriately-configured input method **64**. Indeed, if an optional keyboard **36** is present, keystrokes are directly provided by a keyboard driver **62** to the graphical windowing environment **60**, whereby appropriate keystrokes are likewise placed in the message queue of the active application's window without the application being provided with information as to the source.

Input methods **64** may include, for example, various different displayable keyboards, (soft keyboards), a calculator, a formula and/or equation editor, chemical symbol template, voice recognition, handwriting recognition, shorthand symbol recognition (such as "Graffiti"), or other application-optimized input methods (e.g. a barcode reader). The SIP manager **58** provides a user interface for permitting a user to toggle a SIP window (panel) **50** (FIG. **7**) between an opened and closed state, as described in more detail below. The SIP manager **58** also provides a user interface enabling user selection from a displayable list of available input methods. A user interacting with the user interface may select an input method **64**, and in response, the SIP manager **58** loads and calls the selected input method **64**. In a preferred embodiment, each of the input methods communicates with the SIP manager **58** through a COM (Component Object Model) interface shown as IIMCallback **61** and IInputmethod **63**. A COM object comprises a data structure having encapsulated methods and data that are accessible through specifically defined interfaces. A detailed description of COM objects is provided in the reference entitled "Inside OLE," second edition, Kraig Brockschmidt (Microsoft Press), hereby incorporated by reference.

Generally, when the SIP window **50** is toggled between on/off by a user, as will be described in more detail below, the SIP manager **58** informs the selected input method **64** to correspondingly open/close the SIP window **50** through the IInputmethod mechanism **63**. When a new input method is selected, the SIP manager **58**, through the mechanism **63**, informs any of the previously selected input methods to exit, and loads the newly selected input method. The mechanism **63** may also be utilized by the SIP manager **58** to obtain information specific to a selected input method, as also described in detail below.

The selected input method **64** may also communicate information to the SIP manager **58** via the IIMCallback mechanism **61**, such as which character or characters were entered by a user, irrespective of whether the character or characters are generated through keyboard selection, handwriting recognition, voice recognition, a formula editor, calculator or the like. Such character input is generally passed to the SIP manager **58**, preferably received as (or converted to) a Unicode character (for Windows CE) by the SIP manager **58** and output to the graphical windowing environment **60**. Command key information, such as "Ctrl" on a keyboard, may also be provided by the input method **64** to the SIP manager **58** via interface **61**.

SIP and input method-specific information may also be communicated through the SIP manager **58**, and ultimately to the focused application **29**, when the application is optimized for operating with a SIP (i.e., is "SIP-aware") as described in more detail below.

The system operates as generally represented in the steps of FIG. **3**. Once an application is selected and has focus (steps **300**-**302**), an input method **64** is selected therefor at step **304**. Note that the input method **64** may be selected by the user, or a default input method may be selected for use with a particular application. Additionally, the input method **64** may be one that remains after having been selected for a previous application, i.e., a particular input method stays the same as the user switches between various applications. In any event, the input method **64** displays a SIP window **50** when selected.

As the user inputs data at step **306**, appropriate data is passed to the SIP manager **58** via the IIMCallback mechanism **61**, described below. Note that the input method **64** may first process the received data at step **306**. By way of example, one particular input method **64** may convert barcode symbols to Unicode characters representing digits, another input method may convert mathematical entries into a Unicode result (e.g., an entry of '3+6=' sends a '9' to the SIP manager **58**), while yet another may be an equation editor (e.g., the characters "Sqrt" are converted into a single Unicode value representing a square root symbol). After any such processing, the input method **64** passes those digits to the SIP manager **58**, which in turn passes those digits to the graphical windowing environment **60**. The application receives the character data from the graphical windowing environment **60** as if the user had entered those digits on a physical keyboard, regardless of the input method used.

As shown in FIGS. **5**-**7**, the soft input panel (SIP) functionality of the system collectively includes the visible window **50** (FIG. **7**), a visible SIP button **52**, and various methods and functions (described below). As shown in FIG. **7**, the SIP window **50** is a rectangular area provided by the input method **64** that can be hidden or shown at the user's (or an application program's) request. The visible SIP button **52** is located on a taskbar **56** or the like, and provides a touch-sensitive interface by which the user displays or hides the SIP window **50**. Thus, as represented in the state diagram of FIG. **4**, the window **50** toggles between an open, visible state (FIG. **7**) and a closed, hidden state (FIG. **5**) as the user taps the SIP button **52**. A present design implements a 240 pixel wide by 80 pixel high SIP window **50** that is fixed (docked) on the display **32** at a position just above the taskbar **56**. As will become apparent below, the soft input panel design supports other SIP window **50** sizes or positions.

To this end, the operating system **28** creates a dedicated thread (the SIP manager **58**) that registers itself as a SIP thread with the Windows CE system. The thread creates the SIP window **50**, performs other SIP initialization, and then enters a message loop to respond to messages and user interface activity in the SIP window **50**. The thread also serves to dispatch messages to an Input Method's window, and calls into the Input Method **64** to permit the Input Method **64** to create windows that will respond as special SIP windows.

The SIP manager thread **58** is given special status by the system. For example, windows created by the SIP manager **58** thread are topmost windows, and ordinarily will not be obscured by other windows, except, e.g., when the taskbar **56** is activated in an auto-hide mode while the SIP window **50** is displayed. In this case, the SIP window **50** remains displayed in its current location and the taskbar **56** is displayed on top of the SIP window **50**. More generally, any user interface element for controlling the SIP may (and should) be placed on top of (rather than underneath) the SIP window **50**, whenever the controlling user interface element and the SIP window **50** overlap.

Moreover, when tapped on, the SIP window **50** (and any child windows thereof such as pushbuttons, text entry fields, scrollbars and the like) will not receive the input focus as would conventional program windows. In this manner, the user may interact with the SIP window **50** without changing the system focus. As can be appreciated, changing the system focus each time the user inputs data into the SIP window **50** would be undesirable. The SIP button **52** will also not cause a change of focus for the same reason, i.e., it is undesirable to cause the window with focus to lose focus by tapping on the SIP button **52** to bring out the SIP window **50**.

In accordance with one aspect of the present invention, the SIP system enables the selective installation of a specified Input Method **64**. As generally described above, each Input Method **64** is an interchangeable component by which the user provides character, text or other user data via the touch-screen display (or some other input device). More particularly, the SIP manager **58** preferably exposes a COM interface that enables the selective installation of Input Methods **64**. The Input Method **64** occupies space inside a SIP window **50** created by the system.

Preferably, the Input Method **64** comprises a Component Object Model (COM) object that implements the IInputMethod interface. Notwithstanding, the Input Method **64** and SIP manager **58** can comprise virtually any components capable of communicating with one other through some mechanism, such as by receiving, responding to, and making function calls.

The Input Method **64** is responsible for drawing in the SIP window **50** and responding to user input in the SIP window **50**. Typically, the Input Method **64** will respond to user input and convert that input into characters which are then sent to the SIP manager **58** via exposed SIP functions. By way of example, one Input Method **64** includes a default QWERTY (alpha) keyboard **66** shown in FIG. **7**. More particularly, this Input Method **64** displays an image of the keyboard **66** on the screen **32**, and converts taps on that keyboard **66** (detected as screen coordinates) into characters which are sent to the SIP manager **58** and thereby to the system. Input Methods may be written by application vendors, and are added to the system using COM component installation procedures.

The user interacts with the Input Method **64** manifested in the visible SIP window **50** to create system input. As best represented by the state diagram of FIG. **4** and as shown in FIG. **6**, the user can select a different Input Method by tapping a SIP menu button **70** on the taskbar **56** that provides a pop-up input method list **72** into the SIP window **50**. The user can also select among available Input Methods via a control panel applet (not shown) or the like. The SIP control panel applets communicate with the operating system **28** using the registry and the exposed SIP-aware functionality described below.

As will be described in detail below, the various components cooperate to expose functions, structures, and window messages that enable system applications **29** to respond to changes in the SIP state. An application **29** that uses this functionality to adjust itself appropriately to SIP changes is considered "SIP-aware." Other applications may be SIP-aware yet choose to retain their original size (and thus be partially obscured by the SIP window **50**) when appropriate. Moreover, and as also described below, there are exposed functions that enable applications to programmatically alter the SIP state.

Notwithstanding, applications **29** need not be aware of the SIP system in order to benefit from the present invention. Indeed, one aspect of the present invention is that applications do not ordinarily recognize whether data received thereby originated at a hardware input device such as the keyboard **36** or via user activity (e.g., contact or proximity detected by the screen **32** and detection circuitry **33**) within the soft input panel window **50**. This enables applications to operate with virtually any appropriate input method, irrespective of whether that application is SIP-aware.

Turning to an explanation of the mechanism that facilitates the operation of an Input Method **64** installed by the SIP manager **58**, a SIP-aware application **29** is notified when the SIP window **50** changes state and what the new, current state of the SIP window **50** is. The state includes whether the status of the SIP window **50** is visible or hidden, whether the SIP window **50** is docked or in a floating condition, and the size and position of the SIP window **50**. As shown in the table below, a data structure (SIPINFO) contains this SIP information:

```
Typedef struct {
    DWORD     cbSize
    DWORD     fdwFlags
    RECT      rcVisibleDesktop
    RECT      rcSipRect
    DWORD     dwImDataSize
    Void *pvImData
} SIPINFO;
```

The cbSize field may be filled in by the application program **29** and indicates the size of the SIPINFO structure. This field allows for future enhancements while still maintaining backward compatibility, and indeed, the size of the SIPINFO structure may be used to indicate the version to the components of the system. The fdwFlags field represents the state information of the SIP window **50**, and can be a combination of three flags. A SIPF_ON flag that is set indicates that the SIP window **50** is visible (i.e., not hidden), while a set SIPF_DOC flag indicates the SIP window **50** is docked (i.e. not floating). A set SIPF_LOCKED flag indicates that the SIP window **50** is locked, i.e., the user cannot change its visible or hidden status. Note that a given implementation may not allow floating or locked SIP windows, however the capability is present within the system.

The rcVisibleDesktop field contains a rectangle, in screen coordinates, representing the area of the screen desktop **68** not obscured by the SIP window **50**. If the SIP window **50** is floating (not docked), this rectangle is equivalent to the user-working area. Full-screen applications wishing to respond to SIP window **50** size changes can generally set their window rectangle data structure ("rect") values to this RECT data structure's values. If the SIP window **50** is docked and does not occupy an entire edge (top, bottom, left or right), then this rectangle represents the largest rectangle not obscured by the SIP window **50**. However, the system may provide available desktop space **68** not included in the RECT data structure.

Next, the rcSipRect field contains the rectangle, in screen coordinates, representing the size and location of the SIP Window **50**. Applications **29** will generally not use this information, unless an application **29** wants to wrap around a floating SIP window **50** or a docked SIP window **50** that is not occupying an entire edge.

The dwImDataSize field contains the size of the data pointed to by the PvImData member, which is the next field, i.e., a pointer to the Input Method-specific data. The data are defined by the Input Method **64**.

Whenever the state of the SIP window **50** changes, i.e., a new Input Method has been selected and/or a visibility, docking or size change has occurred, a message, WM_SETTINGCHANGE, is sent to all top-level windows, as generally represented at step **800** of FIG. **8**. In this manner, an application **29** can adjust itself to the new state of the SIP window **50**, such as by adjusting its size in response to this message. To this end, a flag, SPI_SETSIPINFO, is sent with this message to indicate when SIP information has changed, and another flag, SPI_SETCURRENTIM, when the current Input Method has changed. As shown at step **802** of FIG. **8**, the flag is tested to determine if the message is SIP-related or another type of setting change message (whereby it is handled at step **804**). If SIP-related, for performance reasons, the applications that are not currently active in the foreground cache these SIP changes (steps **806**-**808**). If the application's window is active, the application can adjust its size and/or window (steps **810**-**812**). For example, as shown in FIGS. **5** and **6**, when the SIP window **50** of FIG. **7** is hidden and an active application

29 notified, the application **29** may use the additional desktop space **68** to display more information such as the analog clock faces. Note that an application **29** that has cached a SIP change when inactive can query the current SIP state when activated to subsequently adjust itself in an appropriate manner in accordance with the information that is returned.

To query the SIP manager **58**, another function, SHSipInfo, is provided so that applications **29** can determine information about the SIP window **50** and Input Method **64**. In general, if this function succeeds, the return value will be nonzero, while if this function fails, the return value will equal zero and extended error information will be available via a GetLastError( ) call.

The following table sets forth the structure of this call:

```
SHSipInfo (
      UINT uiAction
      UINT uiParam
      PVOID pvParam
      UINT fwinIni
);
```

The uiAction parameter can include the values SIP_SETSIPINFO, SPI_GETSIPINFO, SPI_SETCURRENTIM and SPI_GETCURRENTIM. SIP_SETSIPINFO indicates that pvParam points to a SIPINFO structure (described above). The cbsize, dwImDataSize and pvImDataSize are filled in before calling the SHSipInfo function. In response to this call, the SIPINFO structure is filled in with the current SIP size, state, and visible desktop rectangle. If both dWImDataSize and pvImData are nonzero, the data size and pointer are sent to the Input Method **64**. If the Input Method **64** is called but does not provide Input Method-specific data, or the format or size of the data passed in is not in a format recognized by the Input Method **64**, then the SHSipInfo function call fails (returns zero). If the size and format are supported by the Input Method **64**, the Input Method **64** fills in the buffer that is pointed to by pvImData with the Input Method-specific data. Typically, an application **29** will set the pvImDataSize to zero and pvImData to NULL.

A uiAction of SPI_SETSIPINFO indicates that pvParam points to a SIPINFO structure. The SIP window **50** size and state are set to the values specified in the SIPINFO structure. Before changing a SIP value, the application **29** should first obtain the current SIP state by calling SHSipInfo with SPI_GETSIPINFO, then change whatever specific SIP state values it wishes to change before making the SPI_SETSIPINFO call. The cbSize field is set to the size of the SIP in the structure, and if both pvImDataSize and pvImData are not zero, the data size and pointer are sent to the Input Method **64**. The SHSipInfo call fails if the Input Method **64** is called and does not allow setting Input Method-specific data, or if the format or size of the passed data is not in a format recognized thereby. If a size and format are supported by the Input Method **64**, the Input Method **64** uses the data to set Input Method-specific information. Typically, an application will set the pvImDataSize to zero and pvImData to NULL.

SPI_SETCURRENTIM indicates that pvParam points to a CLSID structure which specifies the CLSID of the Input Method **64** to which the SIP will switch. If the CLSID is not valid, or if the specified Input Method **64** cannot be loaded, the call fails (return value equals zero) and a default Input Method **64** (e.g., the QWERTY-like keyboard **66**) is loaded.

Lastly, a uiAction of SPI_GETCURRENTIM indicates that pvParam points to a CLSID structure that receives the CLSID of the currently selected Input Method **64**.

The IInputMethod Interface

IInputMethod is the interface implemented by the Input Method **64** components. The SIP manager **58** calls the methods of this interface to notify the Input Method **64** of state changes, and request action and information from the Input Method **64**. In general, if the called method succeeds, a success is returned, and conversely, if the method fails, a failure result is returned. The following table sets forth the method calls available in this IInputMethod interface:

```
Interface IinputMethod : Iunknown
{
   HRESULT Select ( [in] HWND hwndSip );
   HRESULT Deselect( void );
   HRESULT Showing ( void );
   HRESULT Hiding ( void );
   HRESULT GetInfo ( [out] IMINFO *pimi );
   HRESULT ReceiveSipInfo ( [in] SIPINFO *psi );
   HRESULT RegisterCallback ( [in] IIMCallback* pIMCallback );
   HRESULT GetImData ( [in] DWORD dwSize, [out] LPVOID
   pvImData );
   HRESULT SetImData ( [in] DWORD dwSize, [in] LPVOID
   pvImData );
   HRESULT UserOptionsDlg ( [in] HWND hwndParent );
}
```

An Input Method **64** will ordinarily receive a Select( ), GetInfo( ), ReceiveSipInfo( ) and Register Callback( ) method call, in sequence, before rendering the SIP window **50** space or responding to user actions. When the SIP window **50** is displayed (i.e., turned on), Showing( ) will be called by the SIP manager **58**, after which the Input Method **64** issues a WM_PAINT message to render the SIP window **50**.

The Select( ) method is called when the Input Method **64** has been selected into the SIP. The Input Method **64** generally performs any desired initialization in response to this call. The Input Method is responsible for drawing the entire client area of the SIP window **50**, and thus ordinarily creates its windows and imagelists (collections of displayable bitmaps such as customized icons) in response to this call. For example, the window handle of the SIP window **50** is provided to the Input Method **64** as a parameter accompanying this Select( ) method call, and the Input Method normally creates a child window of this SIP window **50**. The Input Method **64** is also provided with a pointer to a value, which is set to nonzero by the Input Method **64** if the method call is successful or zero if not successful.

The Deselect( ) method is called when the Input Method **64** has been selected out of the SIP. The Input Method's window should be destroyed in response to this call, and the Input Method **64** will typically perform any other cleanup at this time.

The Showing( ) method will cause the SIP window **50** to be shown upon return from the call. Note that the SIP window **50** is not visible prior to this call, and that once the SIP window **50** is shown, this window and its children will receive paint messages. Conversely, the Hiding( ) method hides the SIP window **50** upon return from the call. Accordingly, the Showing( ) and Hiding( ) methods are used to toggle the SIP window **50** between its open and closed states.

The GetInfo( ) method is called when the system is requesting information about the Input Method **64**. The information requested includes flags indicating any special properties of the Input Method **64**, the handles of two imagelists which contain masked bitmaps that are to be displayed on the SIP button **52** when that Input Method **64** is active, indices into the specified imagelists, and a rectangle indicating the preferred size and placement of the Input Method **64**. The call includes a parameter, pimi, which is a pointer to a data structure (IMINFO) that the Input Method **64** should fill in with appropriate data. The call also provides a pointer to a value that the Input Method should set to nonzero to indicate success and zero to indicate failure. More particularly, the IMINFO data structure is represented in the following table:

```
Typedef struct {
    DWORD cbSize;
    HIMAGELIST hImageNarrow;
    HIMAGELIST hImageWide;
    Int iNarrow;
    Int iWide;
    DWORD fdwFlags;
    Rect rcSipRect;
} IMINFO;
```

The cbSize field contains the size of the IMINFO structure, and is filled in by the SIP manager **58** prior to calling calling GetInfo( ). The hImageNarrow field is a handle to an image-list containing narrow (16×16) masked bitmaps for the Input Method **64**. Similarly, hImageWide is a handle to the image-list containing wide (32×16) masked bitmaps. The SIP manager **58** displays one of the bitmaps (e.g., on the taskbar **56**) to indicate the Input Method **64** that is currently selected. Note that the SIP manager **58** may use the 16×16 or 32×16 bitmaps at various times depending on how it wishes to display the bitmap.

The iNarrow field is an index into the hImageNarrow imagelist indicating which bitmap of several possible from that (narrow) imagelist should currently be displayed. Similarly, the iwide field is an index into the hImageWide image-list indicating which bitmap from that (wide) image list should currently be displayed. Note that the Input Method **64** can initiate a change of the bitmap displayed in the SIP taskbar button **52** by calling IIMCallback::SetImages (described below).

The fdwFlags field indicates the visible, docked and locked states (SIPF_ON SIPF_DOCKED and SIPF_LOCKED) of the Input Method **64**, as well as any special Input Method flags that may be defined in the future. Note that the SIP state flags are ignored for the GetInfo( ) method, but are used in the SetImInfo callback method as described below.

Lastly, the rcSipRect field describes the size and placement of the SIP rectangle. The sizing and placement information returned from GetInfo( ) may be used by the SIP when determining an initial default size and placement. When used, the SetImInfo callback method (described below) specifies the new size and placement of the SIP window **50**.

The ReceiveSipInfo( ) method provides information to the Input Method **64** about the SIP window, including the current size, placement and docked status thereof. This call is made whenever the user, an application **29** or the Input Method **64** changes the SIP state. When the SIP manager **58** sends this information during Input Method initialization, the SIP manger **58** is informing the Input Method **64** of the default SIP settings. The Input Method **64** can choose to ignore these defaults, however the values given are ones that either the user has selected or values that have been recommended as expected or accepted SIP values for that platform. A pointer to the SIPINFO structure that includes this information is passed with this call.

The RegisterCallback method is provided by the SIP manager **58** to pass a callback interface pointer to the Input Method **64**. In other words, the RegisterCallback method call passes an IIMCallback interface pointer as a parameter to the Input Method **64**, whereby the Input Method **64** can call methods on this interface to send information back to the SIP manager **58** as described below. The Input Method **64** uses the callback interface pointer to send keystrokes to applications **29** via the SIP manager **58** and to change its SIP taskbar button icons **52**.

The GetImData( ) method is called when an application program **29** has asked the SIP for the SIPINFOdata structure and has provided a non-NULL pointer for the pvImData member of the SIPINFO structure. The application **29** will ordinarily cause this call to be made when requesting some special information from the Input Method **64**. Two parameters are passed with this call, dwsize, the size of the buffer pointed to by pvImData, and pvImData, a void pointer to a block of data in the application **29**.

With this call, the application **29** is essentially requesting that the Input Method **64** fill the block with information, wherein the size and format of the data are defined by the Input Method **64**. This call is designed for Input Methods **64** that wish to provide enhanced functionality or information to applications. By way of example, a SIP-aware application may wish to know whether a character was entered by way of the SIP or by some other means. An input method **64** can thus respond to the application's request by filling the block.

The SetImData( ) method is called when an application **29** has set the SIPINFO data structure and has provided a non-NULL pointer for the pvImData member of the SIPINFO structure. The application **29** will ordinarily cause this call to be made when requesting that the Input Method **64** set some data therein. The parameters passed with this call include dwsize, the size of the buffer pointed to by pvImData, and pvImData, a void pointer to a block of data in the application **64**.

The IIMCallback Interface

The Input Method **64** uses the IIMCallback interface to call methods in the SIP manager **58**, primarily to send keystrokes to the current application or to change the icon that the taskbar **56** is displaying in the SIP button **52**. The Input Method **64** ordinarily calls the IIMCallback methods only in response to a call thereto which was received through an IInputMethod method call. In general, if the function succeeds, the return value will be a success HRESULT, while conversely, if the function fails, the return value is a failure HRESULT.

The following table represents the IIMCallback Interface:

```
Interface IIMCallback :
Iunknown
{
    Hresult SetImInfo(
            IMINFO *pimi );
    Hresult SendVirtualKey (
            BYTE bVk,
            DWORD dwFlags );
    Hresult SendCharEvents(
            UINT uVk,
            UINT uKeyFlags,
            UINT uChars,
            UINT *puShift,
            UINT *puChars );
    Hresult SendString(
            BSTR ptrzStr,
            DWORD dwChars );
}
```

The first callback, SetImInfo( ) is called by the Input Method **64** to change the bitmaps shown on the SIP taskbar button **52** representing the current SIP, or to change the visible/hidden state of the SIP window **50**. It is also sent by the Input Method **64** to the SIP manager **58** as a notification when the Input Method **64** has changed the size, placement or docked status of the SIP window **50**. By this mechanism, the various Input Methods **64** are able to alert the SIP manager **58** to these types of changes so that the two remain synchronized. By way of example, an Input Method **64** may wish to have a user interface element which allows the user to toggle between a docked state and a floating state, or between one or more subpanels (e.g. keyboard with buttons to switch to a number and/or symbol panel or international symbol panel). The Input Method **64** uses this call to inform the SIP manager **58** of each change in state.

Although not necessary to the invention, all values passed in the IMINFO structure are used by the SIP manager **58**. Consequently, the Input Method **64** should first determine the current state of the SIP window **50** as provided by the SIP manager **58** in the SIPINFO structure received via a prior ReceiveSipInfo( ) method call, described above. Then, the Input Method **64** should make changes to only those settings in which a change is desired, and pass a full set of values back in the IMINFO structure. The pimi parameter is sent as a pointer to an IMINFO structure representing the new Input Method **64** settings, including the size, placement and state of the SIP window **50** as well as the desired Input Method **64** images.

In response to the SetImInfo( ) call, the SIP manager **58** will show or hide the SIP window **50** as specified in the fdwFlags of the IMINFO structure. However, the SIP manager **58** will not resize or move the SIP window **50** if requested, but will instead update the size and placement information returned to applications **29** when queried. If the specified values represent a change from the current SIP state, the SIP manager **58** will notify applications **29** that the SIP state has changed via a WM_SETTINGCHANGE message, described above.

The SendVirtualKey( ) callback is used by an Input Method **64** to simulate a keystroke for a virtual key, e.g., a character or the like entered via the touch screen display **32** or some other Input Method **64**. The key event will be sent to the window which currently has focus (i.e., the window which would have received keyboard input had a key been pressed on an external keyboard). The SendVirtualKey callback modifies the global key state for the virtual key sent, whereby, for example, an Input Method **64** can use this function to send SHIFT, CONTROL, and ALT key-up and key-down events, which will be retrieved correctly when the application **29** calls the GetKeyState( ) API. The SendVirtualKey callback should be used to send virtual key events that do not have associated characters (i.e., keys that do not cause a WM_CHAR sent as a result of TranslateMessage. Note that WM_CHAR, TranslateMessage and other key-related messages are described in the reference “Programming Windows 95”, Charles Petzold, supra). If character-producing virtual keys are sent via this function, they will be modified by the global key state. For example, a virtual key of VK__5 that is sent when the shift state is down will result in a ‘%’ WM_CHAR message for certain keyboard layouts.

Parameters sent with this callback include bVk, which is the virtual keycode of the key to simulate, and dwFlags. The dwFlags may be a combination of a SIPKEY_KEYUP flag, (used to generate either a WM_KEYUP or WM_KEYDOWN), a SIPKEY_SILENT flag, (the key press will not make a keyboard click even if clicks are enabled on the device), or zero.

The SendCharEvent callback allows an Input Method **64** to send Unicode characters to the window having focus, while also determining what WM_KEYDOWN and WM_KEYUP messages the application **29** should receive. This allows the Input Method **64** to determine its own keyboard layout, as it can associate any virtual key with any characters and key state. In keeping with one aspect of the invention, applications **29** thus see keys as if they were sent from a keyboard (i.e., they get WM_KEYDOWN, WM_CHAR, and WM_KEYUP messages). Thus, unlike the SendVirtualKey( ) function, this function does not affect the global key state. By way of example, with the SendCharEvent callback, the Input Method **64** can determine that the shifted (virtual key) VK_C actually sent the Unicode character 0x5564. The shift state flag (specified in the puShift parameter, described below) that is associated with the first character to be sent determines whether a WM_KEYDOWN or WM_KEYUP is generated.

Parameters include uVk, the virtual keycode sent in the WM_KEYUP or WM_KEYDOWN message generated as a result of this function, and a uKeyFlags parameter, a set of KEY state flags that are translated into the lKEYData parameter received in the WM_CHAR, WM_KEYUP or WM_KEYDOWN messages received by the application **29** as a result of this call. Only the KeyStateDownFlag, KeyStatePrevDownFlag, and KeyStateAnyAltFlag key state flags are translated into the resulting lKeyData parameter. The uChars parameter represents the number of characters corresponding to this key event, while the puShift parameter is a pointer to a buffer containing the corresponding KEY_STATE_FLAGS for each character to be sent. If the KeyStateDownFlag bit is sent, this function generates a WM_KEYDOWN message, otherwise it generates a WM_KEYUP message. Lastly, the puchars parameter is a pointer to a buffer containing the characters to be sent.

An Input Method **64** may use the SendString callback to send an entire string to the window which currently has the focus, whereby a series of WM_CHAR messages are posted to the application **29**. An Input Method **64** would typically use this callback after it has determined an entire word or sentence has been entered. For example, a handwriting recognizer or speech recognizer Input Method **64** will use the SendString callback after it has determined that a full word or sentence has been entered.

Parameters of the SendString callback include ptszStr, a pointer to a string buffer containing the string to send, and dwSize, the number of characters to send. This number does not include the null-terminator, which will not be sent.

As can be seen from the foregoing detailed description, there is provided an improved method system for entering user data into a computer system. The method and system are both efficient and flexible, and function with touch-sensitive input mechanisms. With the system and method, a plurality of applications can receive user input from a common input method, while interchangeable input methods may be selected from among a set thereof for each application. The method and system are cost-effective, reliable, extensible and simple to implement.

While the invention is susceptible to various modifications and alternative constructions, a certain illustrated embodiment thereof is shown in the drawings and has been described above in detail. It should be understood, however, that there is no intention to limit the invention to the specific form disclosed, but on the contrary, the intention is to cover all modifications, alternative constructions, and equivalents falling within the spirit and scope of the invention.

What is claimed is:

1. In a computing environment, a computer-implemented method comprising:

displaying an actuatable icon representative of an input method list that includes one or more selectable input methods for one or more computer programs, wherein each input method is a computer-executable software component distinct from the computer programs;

in response to actuation of the actuatable icon, displaying the input method list;

receiving a selection of an input method from the input method list;

installing an input method component that corresponds to the selected input method, the input method component causing an interactive input panel to be displayed;

receiving input via the interactive input panel; and

providing the input to a computer program of the one or more computer programs as if the information was received via user input received from a hardware input device.

2. The method of claim **1** wherein providing the input to the computer program comprises communicating information representative of the input to a graphical windowing environment.

3. The method of claim **2** wherein communicating the information comprises passing the information to an interface.

4. The method of claim **2** further comprising, communicating the information from the graphical windowing environment to an application program, wherein the computer program includes the application program, wherein the information is provided to the application program in a same manner as if the input was received via a hardware keyboard.

5. The method of claim **2** wherein providing the input to the computer program comprises placing the information, by the graphical windowing environment, in a message in a message queue of the computer program, wherein the message queue capable to receive messages corresponding to input from the selected input method and messages corresponding to input from a hardware input device.

6. The method of claim **1** wherein the selected input method corresponds to a displayed keyboard, and wherein receiving input via the interactive input panel that corresponds to the selected input method comprises receiving information corresponding to a keyboard character entered via the displayed keyboard.

7. The method of claim **1** wherein the selected input method corresponds to a handwriting input area, and wherein receiving input via the interactive input panel that corresponds to the selected input method comprises receiving information corresponding to handwritten data.

8. The method of claim **1** further comprising, hiding the input panel.

9. The method of claim **1** further comprising, docking the input panel.

10. At least one computer-readable medium having computer-executable instructions, which when executed perform the method of claim **1**.

11. At least one computer-readable medium having computer-executable instructions stored thereon, which when executed by a computer system perform steps, comprising:

selecting one of a plurality of executable input methods for supplying user input to the computer system, wherein each executable input method is an interchangeable software component distinct from one or more application programs, each executable input method having a defined interface set such that the executable input method is connectable to the application programs;

opening an input window on a display of the computer system independent of a window of an active application program; and

displaying an interactive input panel in the input window, the interactive input panel corresponding to the selected executable input method such that information corresponding to user input received by the selected executable input method via the interactive input panel is provided to the active application program as if the information was received via user input at a hardware input device.

12. The computer-readable medium of claim **11** further comprising, providing an input panel button on the display of the computer system, the input panel button being responsive to open and to close the input window.

13. The computer-readable medium of claim **11** further comprising, providing a Software Input Panel (SIP) menu button on the display of the computer system, the SIP menu button being actuatable to display a selectable list of the plurality of executable input methods.

14. The computer-readable medium of claim **13** further comprising, receiving a selection of one of the plurality of executable input methods displayed in the list as a selected executable input method, and in response, closing any open input window, and opening a new input window corresponding to the selected executable input method.

15. At least one computer-readable medium having computer-executable instructions, which when executed perform steps, comprising:

presenting icons corresponding to a plurality of input methods available for a computer application, wherein each input method is a computer-executable software component distinct from the computer application;

invoking a selected input method in response to a user selecting an icon corresponding to the selected input method, including presenting an input panel window; and

accepting user data entered in the input panel window for the computer application, wherein the user data is provided to the computer application as if the user data was received from a hardware input device.

16. The computer-readable medium of claim **15** wherein accepting user data includes detecting user interaction with a touch-sensitive display.

17. The computer-readable medium of claim **15** wherein each input method comprises a component object model (COM) object, and wherein the step of invoking the selected input method includes the step of instantiating the COM object.

18. The computer-readable medium of claim **15** further comprising converting the user data to a Unicode character value.

19. In a computing environment, a system comprising,

a manager component stored on one or more computer-readable media and configured:

to manage selection of a selected input method from one or more available stored input methods, wherein each input method is a computer-executable software component distinct from one or more computer programs, and

to send input data corresponding to a user input received at the selected input method to a graphical windowing environment; and

the graphical windowing environment to receive the input data and to send the input data to a computer program of the one or more computer programs, wherein the input data is sent to the computer program as if the input data was received via user input received from a hardware input device.

20. The system of claim **19** wherein the computer program comprises an application program having focus.

21. The system of claim **19** further comprising an input panel window corresponding to the selected input method.

22. The system of claim **21** wherein the selected input method presents an image representing a keyboard on the input panel window.

23. The system of claim **21** wherein the manager component selectively displays and hides the input panel window.

24. The system of claim **21** wherein interaction with the input panel does not cause the input panel window to receive focus.

25. The system of claim **19** where the input method is displayed on a touch-sensitive display screen.

26. The system of claim **19** wherein the manager component transfers information from the computer program to the selected input method.

27. The system of claim **19** wherein the selected input method calls functions in the manager component via a defined interface set.

28. The system of claim **19** wherein the selected input method comprises an object.

29. The system of claim **19** wherein the selected input method draws an input panel in an input panel window displayed in the graphical windowing environment.

30. The system of claim **29** wherein the manager component selectively displays and hides the display of the input panel window.

31. The system of claim **29** wherein the manager component docks the input panel window.

* * * * *

Exhibit C

PATENT NO.	TITLE
4,860,003	Communication System Having a Packet Structure Field
5,142,533	Method for Controlling the Scheduling of Multiple Access to Communication Resources
5,164,986	Formation of Rekey Messages in a Communication System
5,239,294	Method for Authentication and Protection of Subscribers in Telecommunication Systems
5,572,193	Method for Authentication and Protection of Subscribers in Telecommunications Systems
5,272,724	Wideband Signal Synchronization
5,319,712	Method and Apparatus for Providing Cryptographic Protection of a Data Stream in a Communication System
5,329,547	Method and Apparatus for Coherent Communication in a Spread-Spectrum Communication System
5,467,398	A Method of Messaging in a Communication System
5,560,021	A Power Management and Packet Delivery Method for Use in a Wireless Local Area
5,636,223	Methods of Adaptive Channel Access Attempts
5,689,563	Method and Apparatus for Efficient Real-Time Authentication and Encryption in a Communication System
5,822,359	A Coherent Random Access Channel in a Spread-Spectrum Communications System and Method
5,311,516	Paging System Using Message Fragmentation to Redistribute Traffic
6,069,896	Capability Addressable Network and Method Therefor
6,331,972	Personal Data Storage and Transaction Device System and Method
5,495,482	Voice and Data Packet Communication Method and Apparatus
5,357,571	A Method for Point-to-Point Communications within Secure Communication Systems
5,412,722	Encryption Key Management
5,029,183	Packet Data Communication System
5,479,441	Packet Data Communication System
5,519,730	Communication Signal Having a Time Domain Pilot Component
6,236,674	Transceiver Control with Sleep Mode Operation
6,404,772	Voice and Data Wireless Communications Network and Method
6,473,449	High-Data-Rate Wireless Local Area Network
7,143,333	Method and Apparatus for Encoding and Decoding Data
7,493,548	Method and Apparatus for Encoding and Decoding Data
7,165,205	Method and Apparatus for Encoding and Decoding Data